Proceedings

TWENTY-SEVENTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

September 27-29, 1989

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

ENGINEERING LIBRARY
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

PROCEEDINGS

TWENTY-SEVENTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

or renew all Librar
$50.00

Juraj V. Medanic
P. R. Kumar
Conference Co-Chairmen

Conference held
September 27-29, 1989
Allerton House
Monticello, Illinois

Sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

Digitized by the Internet Archive
in 2026 with funding from
University of Illinois Urbana-Champaign

https://archive.org/details/proceedings1989alle_0

621.3811
Al5
1989

Engin.

FOREWORD

The Twenty-Seventh Annual Allerton Conference on Communication, Control, and Computing was held at Allerton House, Monticello, Illinois on September 27-29, 1989. The conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. These *Proceedings* contain the papers presented at the Conference.

We are grateful to Professor N. Pippenger of the University of British Columbia and the IBM Almaden Research Center for his plenary lecture ''The Complexity of Switching Networks.'' We are also grateful to Professor S. I. Marcus of the University of Texas for organizing the invited session ''Discrete Event Dynamical Systems.'' We would like to thank all the session chairs and the authors for their contributions.

We appreciate the support of Professor W. K. Jenkins, Director of the Coordinated Science Laboratory and Professor T. N. Trick, Head of the Department of Electrical and Computer Engineering. We would also like to thank our colleagues Professors K. S. Arun, A. Barron, M. T. Başar, T. Ü. Başar Y. Bresler, D. J. Brown, B. E. Hajek, T. S. Huang, D. Jones, P. V. Kokotovic, H. Lee, M. C. Loui, S. Meyn, D. C. Munson, W. R. Perkins, K. R. Poolla, H. V. Poor, F. P. Preparata, M. B. Pursley, D. V. Sarwate and M. W. Spong for their generous assistance in reviewing papers and organizing the papers into sessions.

We are grateful to Mr. Steven Williams for his handling of the business affairs of the Conference, to Mr. Robert Ebeling and his assistants for their timely printing of large numbers of programs and registration packets, to Jennifer Eickman and her fantastic staff at Allerton House who cheerfully accommodated our group, and to Ms. Francie Bridges, Ms. Dixie Murphy, Ms. Janet Reese and Ms. Lila Rhoades in handling the Conference announcement, registration, program preparation, and editing and sales of these *Proceedings*. They are the real organizers of this conference.

Finally, we would like to thank Ms. Christina Kamra of John Wiley & Sons for sponsoring a book display.

J. V. Medanic

P. R. Kumar

ser.

TABLE OF CONTENTS

FOREWORD i

I-A: COMMUNICATION SYSTEMS

Chairman: J. S. Lehnert
Purdue University

A QUADRATIC GAUSSIAN MODEL FOR NONCOHERENT SPREAD-SPECTRUM MULTIPLE-ACCESS COMMUNICATIONS 1
A. R. Akinniyi and J. S. Lehnert

ON THE OPTIMALITY OF DIRECT-SEQUENCE FOR ARBITRARY INTERFERENCE REJECTION 11
M. Hizlan and B. Hughes

JOINTLY OPTIMAL RECEIVERS: AN APPLICATION TO JOINT ACQUISITION AND DETECTION FOR DS-SS SYSTEMS 13
C. N. Georghiades

PERFORMANCE OF NONCOHERENT FREQUENCY HOPPED M-ARY SHIFT KEYING (FH-MFSK) COMMUNICATIONS OVER FACTORY RADIO CHANNEL 15
P. Yegani and C. D. McGillem

A MULTITONE FH/MFSK SYSTEM ON PARTIAL BAND INTERFERENCE AND NOISE 25
G. E. Atkin and H. P. Corrales

PROBABILITY OF ERROR BOUNDS FOR MINIMUM SHIFT KEYING AND GAUSSIAN MINIMUM SHIFT KEYING MODULATION IN FREQUENCY SELECTIVE CHANNELS 27
A. S. Amanullah

MULTIPLE ACCESS SCHEMES WITH CONTENTION 29
S. M. Lee and D. W. Browning

ON THE ERROR PROBABILITY OF SIGNALS IN ADDITIVE GAUSSIAN NOISE........ 39
B. Hughes

I-B: GRAPH ALGORITHMS

Chairman: C. L. Lin
University of Illinois at Urbana-Champaign

TESSELLATION REPRESENTATIONS OF PLANAR GRAPHS 48
R. Tamassia and I. G. Tollis

CONSTRAINED PLANAR EMBEDDING PROBLEMS 58
M. F. M. Stallmann

AN EFFICIENT ALGORITHM FOR DOMINATION IN PERMUTATION GRAPHS 68
C. Rhee, S. K. Dhall, S. Lakshmivarahan and H. Kim

TR-MATCHING FOR CHORDAL GRAPHS 72
J. R. S. Blair and S. S. Ravi

GENERATION OF MAXIMUM INDEPENDENT SETS OF A BIPARTITE GRAPH AND MAXIMUM CLIQUES OF A CIRCULAR-ARC GRAPH 82
T. Kashiwabara, S. Masuda, K. Nakajima and T. Fujisawa

RELATIONSHIP BETWEEN DENSITY AND TOTAL EDGE LENGTH IN BIPARTITE ARRANGEMENT 92
T. N. Bui, S.-L. Lee, and K.-P. Liou

A LINEAR TIME ALGORITHM FOR VERTEX-DELETION BIPARTIZATION ON OUTERPLANAR GRAPHS 102
H.-A. Choi

I-C: LINEAR SYSTEMS

Chairman: Tamer Başar
University of Illinois at Urbana-Champaign

FILTRATIONS FOR LINEAR SYSTEMS 104
B. F. Wyman and M. K. Sain

DISCRETE-TIME p-POWERS AND STABILITY 111
I. W. Sandberg

ROBUST PERFORMANCE IN THE PRESENCE OF ADDITIVE AND MULTIPLICATIVE UNCERTAINTIES 118
J. Chen and J. S. Freudenberg

ON THE SPECTRAL SET FOR A FAMILY OF POLYNOMIALS 128
B. R. Barmish and R. Tempo

ON THE STABILITY OF POLYGONS OF POLYNOMIALS 136
L. R. Pujara and N. R. Shanbhag

ALGEBRAIC VS ANALYTIC DESIGN LIMITATIONS IMPOSED BY ILL-CONDITIONED PLANTS 138
J. S. Freudenberg

NEAR-OPTIMUM STEADY STATE REGULATORS FOR STOCHASTIC LINEAR WEAKLY COUPLED DISCRETE SYSTEMS 148
X. Shen and Z. Gajic

I-D: NONLINEAR SYSTEMS

Chairman: Mark Spong
University of Illinois at Urbana-Champaign

NONLINEAR OUTPUT REGULATION: REMARKS ON ROBUSTNESS 150
C. I. Byrnes and A. Isidori

A METHOD FOR HIGHER ORDER, DYNAMICAL FEEDBACK IN NONLINEAR SERVOMECHANISM PROBLEMS 159
P. M. Sain, M. K. Sain and A. N. Michel

THE POSTURAL STABILITY IN BIPED DURING SITTING DOWN MOTION 169
H. D. Ong, H. Hemami and S. Simon

REDUNDANCY RESOLUTION AND CONTROL OF REDUNDANT MANIPULATORS USING REDUCED-ORDER MODEL 179
S. H. Arnautovic and A. J. Koivo

A MODELING MECHANISM FOR HYBRID INTERACTING DISCRETE-EVENT AND CONTINUOUS-TIME SYSTEMS 181
W. H. Cheong and T. L. Casavant

REDUCED ORDER OBSERVER DESIGN FOR DISCRETE STOCHASTIC BILINEAR SYSTEMS 191
E. Yaz

II-A: ADAPTIVE SYSTEMS AND IDENTIFICATION

Chairman: R. H. Middleton
University of Illinois at Urbana-Champaign

ASYMPTOTIC PROPERTIES OF PARAMETER ESTIMATION FOR THE DMC MODELS 199
O. A. Asbjornsen, K. Yin and G. Yin

ADAPTIVE CONTROL OF A CLASS OF NONMINIMUM PHASE PLANT PART I: PROOF OF STABILITY 209
S. Bai

RECURSIVE ESTIMATION USING PIPELINE STRUCTURE 219
G. Yin

NOMINAL ADAPTIVE CONTROL FOR A ROBOT MANIPULATOR 229
D. M. Dawson and J. F. Dorsey

REAL-TIME VISION SERVOING OF A MANIPULATOR WITH SELF-TUNING CONTROLLER 231
N. Houshangi and A. J. Koivo

A COMPARISON OF ARMA AND IMPULSE IDENTIFICATION 233
M. S. Moellenhoff and K. T. Erickson

STABILITY ANALYSIS OF A MULTIVARIABLE MODEL-BASED PREDICTIVE CONTROLLER 243
A. D. Raney and K. T. Erickson

ADAPTIVE MARKOV DECISION PROBLEMS UNDER CONSTRAINTS 253
E. Altman

II-B: COMBINATORIAL ALGORITHMS

Chairman: Martin L. Brady
Lockheed

ON THE MAXIMUM QUEUE LENGTH WITH APPLICATIONS TO DATA STRUCTURES ANALYSIS 263
W. Szpankowski

ON THE CONSTRUCTION OF MINIMUM HEIGHT GENERALIZED BINARY TREES 273
V. Viswanathan and S.-H. S. Huang

INTERPOLATION SEARCH FOR ALPHABETIC TABLES 282
Y. Perl and L. Gabriel

AN EFFICIENT ALGORITHM FOR A SPECIAL CASE OF THE SET PARTITION PROBLEM 284
A. Jackson and P. Ramanan

A LINEAR TIME ALGORITHM FOR THE TWO TRACK ASSIGNMENT PROBLEM 291
W.-L. Hsu and K.-H. Tsai

SWITCHBOX ROUTING FOR MULTITERMINAL NETS IN MANHATTAN MODE 301
M. Kaufmann and I. G. Tollis

RECONFIGURABILITY AND WIRABILITY OF PLANAR LAYOUTS 310
S.-Q. Zheng

II-C: CODING THEORY

Chairman: D. J. Costello, Jr.
University of Notre Dame

ON THE PERFORMANCE OF MULTIDIMENSIONAL PHASE MODULATED TRELLIS CODES 312
L. C. Perez and D. J. Costello, Jr.

VITERBI DECODER PATH MEMORY AND MULTIDIMENSIONAL TRELLIS CODED MODULATION 322
J. M. Nowack and M. A. Herro

SOME NEW METHODS FOR SIMULATING SEQUENTIAL DECODERS USING IMPORTANCE SAMPLING 324
K. B. Letaief and J. S. Sadowsky

A SYMBOL ERROR UPPER BOUND FOR CONVOLUTIONAL CODES 334
R. J. McEliece and I. M. Onyszchuk

A PROGRAMMABLE SYSTOLIC BCH DECODER 336
J. J. Vaccaro, T. M. Hopkinson and K. C. Moore

GENERALIZED DE BRUIJN SEQUENCES 346
R. J. McEliece and L. Popovic

SPECTRALLY CONSTRAINED M-ARY ERROR CORRECTING CODES WITH APPLICATION TO QAM SYNCHRONISATION 355
J. J. O'Reilly and A. Kokkos

II-D: ANALYSIS OF QUEUEING SYSTEMS

Chairman: Bruce Hajek
University of Illinois at Urbana-Champaign

THE QUEUE IN A STATISTICAL MULTIPLEXER, AS SEEN BY AN ARRIVING PACKET 364
F. M. Brochin and J. B. Thomas

TRANSIENT AND EQUILIBRIUM ANALYSIS OF SPREAD SPECTRUM SLOTTED ALOHA NETWORKS: FINITE MEMORY RECURSIONS 374
G. R. Murthy and E. J. Coyle

SOME EFFICIENT COMPUTATIONAL ALGORITHMS RELATED TO PHASE MODELS 383
V. Rego

A DECOMPOSITION ALGORITHM FOR THE STATE TRANSITION LATTICES OF CERTAIN NON-PRODUCT QUEUEING PROTOCOLS 385
C.-K. Hong and I. Y. Wang

A NEW ANALYSIS OF THE SELECTIVE-REPEAT ARQ PROTOCOL 395
S.-J. Wey

INPUT BUFFER REQUIREMENTS FOR ROUND ROBIN POLLING SYSTEMS 397
G. Sasaki

MAXIMUM QUEUES IN M/M/1 AND M/D/1 SYSTEMS 407
W. Furth and N. Duchon

II-E: SYTEMS THEORY

Chairman: Joseph Bentsman
University of Illinois at Urbana-Champaign

RELEGATION FOR DECENTRALIZED CONTROL OF COUPLED SYSTEMS 421
D. A. Schoenwald and Ü. Özgüner

ROBUST CONTROL SYSTEM DESIGN BASED ON PROBABILISTIC DESCRIPTIONS OF PLANT UNCERTAINTY 429
C. D. Schaper, D. A. Mellichamp and D. E. Seborg

A C-A-D PACKAGE FOR LTR DESIGN VIA ASYMPTOTIC EIGENSTRUCTURE AND TIME-SCALE STRUCTURE ASSIGNMENT 439
B. M. Chen, S. Bingulac and A. Saberi

PERFORMANCE AND STABILITY ROBUSTNESS OF LEB (LINEAR-ELLIPSOIDAL-BOUNDED) COMPENSATORS 441
C. D. Antonini

SOME APPLICATIONS OF INTERVAL ANALYSIS METHODS TO CIRCUITS AND SYSTEMS 451
A. N. Michel and H. F. Sun

SYSTOLIC ARRAYS FOR SOLVING LINEAR TIME INVARIANT SINGULAR SYSTEMS 461
A. K. ElMusrati and M. E. Zaghloul

AN ALGEBRAIC METHOD FOR DETERMINING AND COUNTING ALL REGIONS IN A LINEAR PARTITION OF R^n 471
C. Güzelis and İ. C. Göknar

III-A: DISCRETE EVENT DYNAMICAL SYSTEMS

Organizer: S. I. Marcus
University of Texas
Chairman: P. Ramadge
Princeton University

THE PROTOCOL CONVERSION PROBLEM - FINDING A QUOTIENT OF SPECIFICATIONS 481
K. L. Calvert and S. S. Lam

A NOTE ON SUPREMAL CONTROLLABLE AND NORMAL SUBLANGUAGES 491
F. Lin, R. D. Brandt and W. M. Wonham

SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS: SUPREMAL CONTROLLABLE AND RECOGNIZABLE LANGUAGES 501
R. Kumar, V. Garg and S. I. Marcus

HIERARCHICAL CONTROL OF DISCRETE EVENT SYSTEMS: COMPUTATION AND EXAMPLES 511
H. Zhong and W. M. Wonham

ASYNCHRONOUS COORDINATION PROBLEMS FOR DISCRETE EVENT SYSTEMS 520
C. H. Golaszewski and P. J. Ramadge

LINEAR INTEGER PROGRAMMING TECHNIQUES IN THE CONTROL OF VECTOR DISCRETE-EVENT SYSTEMS 528
Y. Li and W. M. Wonham

III-B: COMPUTATIONAL GEOMETRY

Chairman: F. P. Preparata
University of Illinois at Urbana-Champaign

COOPERATIVE SUBDIVISION SEARCH ALGORITHMS WITH APPLICATIONS 538
N. Dadoun and D. G. Kirkpatrick

EUCLIDEAN SHORTEST PATH IN THE PRESENCE OF OBSTACLES 548
Y.-M. Chen and P. Ramanan

SOME APPLICATIONS OF THE GEOGRAPHIC NEAREST NEIGHBOR APPROACH 555
Y. C. Wee, S. Chaiken and S. S. Ravi

OUT-OF-ROUNDNESS PROBLEM REVISITED 564
V. B. Le and D. T. Lee

FINDING A MINIMAL CHAIN TO SEPARATE TWO POLYGONS 574
C. A. Wang

AN OPTIMUM ALGORITHM FOR COVERING A HORIZONTALLY CONVEX ORTHOGONAL POLYGON WITH ORTHOGONAL STAR-SHAPED POLYGONS 584
L. Gewali and S. Ntafos

III-C: COMMUNICATION NETWORK ALGORITHMS

Chairman: G. H. Sasaki
University of Texas at Austin

A LOOP-FREE SPANNING-TREE PROTOCOL IN DYNAMIC TOPOLOGY 594
C. Cheng and S. P. R. Kumar

A FAST TOPOLOGY MAINTENANCE ALGORITHM FOR HIGH-BANDWIDTH NETWORKS 596
H. Abu-Amara

DATA REDISTRIBUTION SCHEDULING IN FULLY CONNECTED NETWORKS 603
R. Varadarajan, P. I. Rivera-Vega and S. B. Navathe

OPTIMAL ALGORITHMS FOR ALL TO ALL BROADCAST 613
A. Bagchi, S. L. Hakimi and E. F. Schmeichel

EDGE-LABELLED GOSSIP IN ROUNDS 621
A. L. Liestman and D. Richards

TIME-SPACE TRADEOFFS IN NONBLOCKING PARALLEL COMMUNICATIONS 631
D. M. Koppelman and A. Y. Oruç

ON SELF-ROUTABILITY OF THREE-STAGE SWITCHING NETWORKS 640
B. G. Douglass and A. Y. Oruç

III-D: ESTIMATION AND DETECTION

Chairman: James Bucklew
Princeton University

RANDOMIZATION AND SCHEDULING IN DECENTRALIZED DETECTION 649
P. Willett and D. Warren

DENSITY ESTIMATION IN CORRELATED NOISE 651
T. J. Schulz and D. L. Snyder

THE CR BOUNDS FOR JOINT ESTIMATES OF INDEPENDENT AND PROPORTIONAL FM SIGNALS 653
T. E. Luginbuhl and P. K. Willett

A SIEVE-CONSTRAINED MAXIMUM-LIKELIHOOD ESTIMATOR FOR THE SPECTRUM OF A GAUSSIAN PROCESS 663
P. Moulin, D. L. Snyder and J. A. O'Sullivan

IDENTIFIABILITY OF AR PARAMETERS OF NONCAUSAL ARMA SIGNALS FROM OUTPUT CUMMULANTS 672
J. K. Tugnait

SIGNAL DETECTION USING THE BISPECTRUM 682
P. A. Nielsen and J. B. Thomas

ON AN ASPECT OF OPTIMAL NONLINEAR ESTIMATION 692
E. B. Hall and G. L. Wise

III-E: COMPUTATIONAL MODELS AND COMPLEXITY

Chairman: David Muller
University of Illinois at Urbana-Champaign

SEQUENTIAL COMPUTATION BASED ON PIPELINED ACCESS TO MEMORY 702
F. Luccio and L. Pagli

NORMALIZED TIME AND ITS USE IN ARCHITECTURAL DESIGN 712
S. Ho, T. Holman and L. Snyder

A LOGIC FOR PETRI NETS AND ITS APPLICATIONS 714
H.-C. Yen

FINITE AUTOMATON APERIODICITY IS PSPACE-COMPLETE 724
S. Cho and D. T. Huynh

ON THE COMPLEXITY OF USING BDDs FOR THE SYNTHESIS AND ANALYSIS OF BOOLEAN CIRCUITS 730
S. Chakravarty

ORDERING STRUCTURES AND THE KNUTH-BENDIX COMPLETION ALGORITHM 740
M. Okada and A. Steele

IV-A: PARALLEL ALGORITHMS

Chairman: Pravin Vaidya
University of Illinois at Urbana-Champaign

PARALLEL BINARY SEARCH 748
S. G. Akl and H. Meijer

PARALLEL TESTING FOR THE CONSECUTIVE ONES PROPERTY AND TRANSFORMABLE CONVEX BIPARTITE GRAPHS AND FINDING A MAXIMUM MATCHING 756
L. Chen and Y. Yesha

OPTIMAL PARALLEL ALGORITHMS ON SORTED INTERVALS 766
S. K. Kim

AN NC ALGORITHM FOR FINDING A MAXIMUM WEIGHT CLIQUE ON A CIRCLE GRAPH 776
C. Rhee, S. K. Dhall and S. Lakshmivarahan

PARALLEL GARBAGE COLLECTORS FOR THE RP3 778
P. Revesz

PARALLEL ALGORITHMS FOR SCHEDULING TASKS WITH PRECEDENCE CONSTRAINTS ON A SINGLE MACHINE 788
E. Dekel and C.-H. Shyu

PARALLEL RANDOM BIT GENERATION ON SIMD ARCHITECTURES 798
M. L. Brady

IV-B: OPTIMIZATION OF QUEUEING SYSTEMS

Chairman: S. P. R. Kumar
Northwestern University

BULK-SERVICE QUEUE WITH SERVICE DELAY POLICY 800
R. W. Yeung

OPTIMAL SCHEDULING OF PARALLEL PROCESSING SYSTEMS WITH REAL-TIME CONSTRAINTS 808
F. Baccelli, Z. Liu and D. Towsley

A FAMILY OF JOB SCHEDULING STRATEGIES FOR AVERAGE RESPONSE TIME CONSTRAINTS 810
L. Georgiadis, C. Nikolaou, and I. Viniotis

OPEN-LOOP FEEDBACK POLICIES FOR MARKOVIAN QUEUEING SYSTEMS 820
R. A. Milito

OPTIMAL DISTRIBUTED ROUTING ALGORITHMS FOR DATAGRAM COMMUNICATION NETWORKS 829
J. R. Yee

STOCHASTIC APPROXIMATION SCHEMES FOR A LOAD BALANCING PROBLEM 839
R. Simha and J. F. Kurose

ADAPTIVE ROUTING & CONGESTION CONTROL FOR WINDOW FLOW CONTROLLED VIRTUAL CIRCUIT NETWORKS 849
A. A. Economides, P. A. Ioannou and J. A. Silvester

IV-C: ROBUST STABILIZATION

Chairman: K. Poolla
University of Illinois at Urbana-Champaign

STABILIZATION AND EXACT TRACKING OF FAST MOVING TIME-VARYING SYSTEM 858
P. R. Bouthellier and B. K. Ghosh

INDIRECT REPETITIVE CONTROL FOR LINEAR DISCRETE MULTIVARIABLE SYSTEMS 867
M. Phan, R. W. Longman and J.-N. Juang

ROBUSTNESS OF COVARIANCE CONTROLLERS 877
G. Zhu, M. Corless and R. Skelton

ROBUSTNESS OF DISCRETE-TIME SYSTEMS IN THE PRESENCE OF UNSTRUCTURED STOCHASTIC PERTURBATIONS 884
E. Yaz

ROBUST LINEAR STATE FEEDBACK STABILIZATION OF CONTINUOUS TIME UNCERTAIN DYNAMIC SYSTEMS 886
Z. Qu and J. Dorsey

DISTRIBUTED SENSING FOR ROBUST CONTROL OF FLEXIBLE STRUCTURES 895
M. F. Barsky and D. K. Lindner

SLIDING MODE CONTROL FOR SINGULARLY PERTURBED SYSTEMS 903
B. S. Heck

STABILITY REGIONS OF UNCERTAIN SYSTEMS WITH BOUNDED CONTROLLERS 913
M. Madani-Esfahani, M. Hached and S. H. Żak

IV-D: DISCRETE EVENT SYSTEMS

Chairman: S. I. Marcus
University of Texas at Austin

NEAR-OPTIMAL CONTROL OF DISCRETE EVENT SYSTEMS 915
K. M. Passino and P. J. Antsaklis

SUPERVISORY CONTROL OF VECTOR DISCRETE-EVENT SYSTEMS 925
N. Q. Huang, Y. Li and W. M. Wonham

COMMENTS ON THE SENSITIVITY OF THE OPTIMAL COST AND THE OPTIMAL POLICY FOR DISCRETE MARKOV DECISION PROCESSES 935
E. L. Sernik and S. I. Marcus

OBJECT-ORIENTED COMPUTATION AND SIMULATION OF LARGE-SCALE DISCRETE-EVENT SYSTEMS 945
S. D. O'Young and W. M. Wonham

IV-D-2: SAMPLE DATA SYSTEMS

Chairman: W. R. Perkins
University of Illinois at Urbana-Champaign

STABILITY OF SAMPLED-DATA FEEDBACK SYSTEMS 955
T. Chen and B. A. Francis

ASYMPTOTIC STATISTICAL PROPERTIES OF LINEAR SYSTEMS OPERATING UNDER QUANTIZED FEEDBACK 965
D. F. Delchamps

MOMENT MATCHING MODEL REDUCTION FOR MULTIRATE LINEAR SYSTEMS 975
D. Williamson, R. E. Skelton and G. Zhu

IV-E: DIGITAL SIGNAL AND IMAGE PROCESSING

Chairman: Y. Bresler
University of Illinois at Urbana-Champaign

ANALYSIS AND RECOVERY OF SAMPLE-AND-HOLD SIGNAL WITH IRREGULAR SAMPLES 977
F. Marvasti

SPECTRAL FILTERING ANALYSIS OF THE CONJUGATE GRADIENT METHOD APPLIED TO SIGNAL EXTRAPOLATION 979
B. J. Sullivan

THE CLASSIFICATION AND ASSOCIATIVE MEMORY CAPABILITY OF STACK FILTERS 981
P.-T. Yu and E. J. Coyle

NON-LINEAR DESCRIPTION OF SIGNALS USING MATHEMATICAL MORPHOLOGY 991
A. Morales and R. Acharya

SHAPE RECOGNITION BY MAXIMUM LIKELIHOOD METHODS 993
K.-B. Eom and J. Park

MOTION FIELD SEGMENTATION FOR IMAGE SEQUENCE CODING 1003
M. Orchard

MOTION COMPENSATION CODING AUGMENTED BY INTRAFRAME PREDICTION 1005
D. F. Wilkins and N. R. Cox

MULTIPLE-FRAME FORMULATION OF THE PEL-RECURSIVE MOTION ESTIMATION ALGORITHM 1007
S. N. Efstratiadis and A. K. Katsaggelos

PLENARY LECTURE

EXACTLY SOLVABLE TELEPHONE SWITCHING PROBLEMS 1017
N. Pippenger
University of British Columbia and the IBM Almaden Research Center

V-A: NEURAL NETWORKS

Chairman: S. P. Meyn
University of Illinois at Urbana-Champaign

ANALYSIS AND SYNTHESIS OF A CLASS OF DISCRETE-TIME NEURAL NETWORKS DESCRIBED ON HYPERCUBES 1027
A. N. Michel, J. Si and G. Yen

ARTIFICIAL NEURAL NETWORKS IN THE MATCH PHASE OF RULE-BASED EXPERT SYSTEMS 1037
M. A. Sartori, K. M. Passino and P. J. Antsaklis

ANALYSIS OF BACK PROPAGATION ALGORITHM FOR FEED-FORWARD NEURAL NETWORKS 1047
H. Guo and S. B. Gelfand

PERIODIC AND PULSE FUNCTION GENERATION USING A NEURAL NETWORK 1049
D. L. Gray and A. N. Michel

AN EFFICIENT NEURAL NETWORK REALIZATION OF NONLINEAR FUNCTIONS 1051
Q. Huang and R.-W. Liu

TRAINING NEURAL NETS FOR ROBUST SHAPE CLASSIFICATION 1060
L. Gupta, M. R. Sayeh and R. Tammana

V-B: DISTRIBUTED AND PARALLEL COMPUTING

Chairman: M. C. Loui
University of Illinois at Urbana-Champaign

A BYZANTINE AGREEMENT PROTOCOL WITH OPTIMAL MESSAGE BIT COMPLEXITY 1062
B. A. Coan and J. L. Welch

INEFFECTIVENESS OF SYNCHRONOUS COMMUNICATION 1072
R. Condamoor and S. Venkatesan

AN INTERVAL LOGIC FOR REASONING ABOUT TIMING PROPERTIES OF COMPUTER COMMUNICATION PROTOCOLS 1074
M.-J. Sheu, C.-K. Chang and C.-L. Lei

PARALLEL FRACTIONAL CASCADING ON A HYPERCUBE MULTIPROCESSOR 1084
F. Dehne, A. Ferreira and A. Rau-Chaplin

ALGORITHMIC UTILISATION OF LOCAL MEMORY IN MESH CONNECTED COMPUTERS ... 1094
N. Blais and H. ElGindy

V-C: INFORMATION THEORY

Chairman: A. Barron
University of Illinois at Urbana-Champaign

CAPACITY FOR A CLASS OF ADDITIVE DIMENSION-LIMITED CHANNELS 1103
C. R. Baker

TOTAL CAPACITY OF THE RMS BANDLIMITED K-USER PAM SYNCHRONOUS CHANNEL . 1112
R. S. Cheng and S. Verdu

CODING CAPACITY OF THE POISSON CHANNEL WITH TIME-VARYING PARAMETERS 1122
M. R. Frey

V-D: STATISTICAL SIGNAL PROCESSING

Chairman: K. S. Arun
University of Illinois at Urbana-Champaign

MULTIPLE SOURCE SIGNALS SEPARATION IN NOISE 1132
L. Tong, V. C. Soon, Y. F. Huang and R. W. Liu

A FAST COVARIANCE LEAST SQUARES ALGORITHM FOR FREQUENCY ESTIMATION 1142
R. M. Todd and J. R. Cruz

A GENERALIZED SIGNAL CLASSIFIER BASED ON A PROBABILISTIC INTERPRETATION OF SPECTRAL CORRELATION 1152
Y. Kim and C. L. Weber

ROBUST DIRECTION-OF-ARRIVAL ESTIMATION 1162
D. D. Lee and R. L. Kashyap

CONVERGENCE ANALYSIS OF THE LVCMS ALGORITHM FOR UNCORRELATED GAUSSIAN DATA 1172
S. H. Nam and J. D. Gibson

FIXED-POINT IMPLEMENTATION OF A LEAST SQUARES EQUALIZER USING HIGHLY QUANTIZED COEFFICIENTS 1174
S. H. Leung and B. L. Chan

INDEX OF AUTHORS 1176

A QUADRATIC GAUSSIAN MODEL FOR NONCOHERENT SPREAD-SPECTRUM MULTIPLE-ACCESS COMMUNICATIONS

AKINGBADE R. AKINNIYI, AND JAMES S. LEHNERT
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907
(317) 494-2362

ABSTRACT

We are proposing a quadratic Gaussian model for the characterization of a noncoherent DS/SSMA communication system. Expressions for the output of the receiver are obtained and used to determine the distribution for the decision statistic. The effects of the length of the signature sequences and the number of interfering transmitters are illustrated. Also, conditions are identified that allow the multiple-access interference to be modeled by a set of independent Gaussian random variables.

I. INTRODUCTION

In recent years, several results have been obtained on noncoherent DS/SSMA systems over additive white Gaussian noise (AWGN) and fading multiple-access channels [1] — [3]. In most publications, the MAI was modeled by a set of independent Gaussian random variables.

Here, we present a characterization of noncoherent DS/SSMA communications without the ordinary Gaussian assumption for the MAI. We utilize conditions obtained in [4] and [5] under which the MAI can be assumed to be conditionally Gaussian. We consider the case involving K users, each of which is assigned a set of signature sequences. All signature sequences are assumed to be randomly generated. Each transmitter is also assumed to have equal power, although the approach can accommodate the case of transmitters with unequal powers. We obtain an expression for the decision statistic, which is modeled as an inhomogeneous quadratic Gaussian random variable after appropriate conditioning. This model allows us to treat any number of users on the channel simply by computing some key parameters described in the Theorem that follows. It is noted that the decision statistic cannot always be adequately modeled as a chi-squared (χ^2) distributed random variable. The dependence of the density function of the decision statistic on the length of the signature sequences is demonstrated.

II. SYSTEM MODEL AND ANALYSIS

The system model that we examine involves K active transmitters and a single typical receiver. The model of the transmitter and the multiple-access channel is depicted in Figure 1. Each of the K users in the system is assigned a pair of randomly generated signature sequences of length N. These sequences are used to produce a pair of wideband signals. In particular, the *k-th* user is assigned the code $V^{(k)}$ described by

$$V^{(k)} = \left\{ V_0^{(k)}, V_1^{(k)} \right\}, \tag{1}$$

where

$$V_\mu^{(k)} = (V_{\mu,0}^{(k)}, V_{\mu,1}^{(k)}, \ldots\ldots\ldots\ldots, V_{\mu,N-1}^{(k)}) \qquad \mu=0,1. \tag{2}$$

The sequence $\{ V_{\mu,j}^{(k)} \}$ of 1's and −1's of length N is used to form the spectral spreading signal $V_\mu^{(k)}(t)$ that is given by

$$V_\mu^{(k)}(t) = \sum_{j=0}^{N-1} V_{\mu,j}^{(k)} \psi(t-jT_c) \qquad \mu=0,1, \tag{3}$$

where $\psi(t)$ is the common chip waveform for all signals. The chip waveform is time limited to the interval $[0, T_c)$, where the chip duration T_c satisfies the relationship $T_c = T/N$. The chip waveform is also normalized such that $\int_0^{T_c} \psi^2(t)dt = T_c$. Whenever the *k-th* user is transmitting a zero, $V_0^{(k)}(t)$ is sent, and when a one is transmitted, $V_1^{(k)}(t)$ is sent. The data stream is modeled as an independent and identically distributed sequence. Each element in the sequence is a zero or a one with equal probability. The transmitted signal waveform is then constructed such that

$$V_k(t) = \sum_{\ell=-\infty}^{\infty} [1-\mu_k(t)]V_0^{(k)}(t-\ell T) + \mu_k(t)V_1^{(k)}(t-\ell T) , \tag{4}$$

where $\mu_k(t)$ is an indicator function for the *k-th* user given by

$$\mu_k(t) = \sum_{\ell=-\infty}^{\infty} b_\ell^{(k)} p_T(t-\ell T) \tag{5}$$

and the ℓ−th data bit $b_\ell^{(k)} \in \{0,1\}$. Also $p_T(t)=1$, for $0 \le t < T$, and $p_T(t)=0$, otherwise. Observe that for $(n-1)T \le t < nT$, $\mu_k(t) = \mu_k((n-1)T)$. For notational convenience when $(n-1)T \le t < nT$, we will write $\mu_k((n-1)T)$ as $\mu_{(n-1)}^{(k)}$. Using the analytic signal representation, the *k-th* transmitter sends $\mathrm{Re}[\, z_k(t)]$, where

$$z_k(t) = \sqrt{2P_s}\, V_k(t)\exp(j(\omega_0+\theta_k)). \tag{6}$$

In the above expressions, ω_0 is the carrier frequency of the bandpass signal of (6). The term θ_k represents the phase of the *k-th* carrier, and P_s is the common signal power. The results that follow can easily be modified for unequal center frequencies and power levels. Finally, the transmitted signal for the *k-th* user undergoes a time delay of τ_k .

The receiver employs two matched filters in which the received signal is correlated over a symbol duration with replicas of the two possible transmitted signals from the first user. Then the filter outputs are processed by an envelope detector and sampled to form two decision variables $\Gamma_0^{(1)}$ and $\Gamma_1^{(1)}$. The larger of the two is selected. The input signal at the receiver is the sum of the delayed and phase-shifted signals from all transmitters, and is given by $\mathrm{Re}[r(t)]$. Thus

$$r(t) = \sqrt{2P_s}\, \{ V_1(t)\exp(j\omega_0 t) + \sum_{k=2}^{K} V_k(t-\tau_k)\exp(j(\omega_0 t + \phi_k)) \} , \tag{7}$$

where $\phi_k = \theta_k - \omega_0\tau_k$. We consider the receiver matched to the first transmitter as our reference. Since we are concerned with relative time delays modulo T, we may assume, without loss of generality that $\tau_1 = 0$ and consider only $0 \le \tau_k < T$ for $k \neq 1$. Similarly, we assume $\phi_1=0$ and have $0 \le \phi_k < 2\pi$ for $k \neq 1$. We are not assuming that the system is coherent by making the assumption that $\phi_1=0$. In our model, the receiver does not use this information during demodulation. Hence, the analysis remains essentially unchanged if we include ϕ_1 as a parameter until it disappears after the envelope detector. The second term in (7) is the multiple access interference (MAI) from other users. We model each time delay τ_k and each phase shift θ_k as random variables that are uniformly distributed on $[0,T)$ and $[0,2\pi)$, respectively. Both random variables are independent of all other random variables in the system and channel model.

The output of the correlation filter to the i-th receiver, denoted by $Z_\lambda^{(i)}(t)$, is given by

$$Z_\lambda^{(i)}(t) = \frac{1}{2}\mathrm{Re}\left[\exp(j\omega_0 t)\int_{t-T}^{t} r(u)V_\lambda^{(i)}(u)\exp(-j\omega_0 u)du\right] \qquad \lambda=0,1. \tag{8}$$

This expression for $\max\{\tau_2,\ldots,\tau_K\}<t<\min\{\tau_2,\ .\ .\ .\ ,\tau_K\}+T$ becomes [4], [5]

$$Z_\lambda^{(i)}(t) = T\sqrt{P_s/2}\,\mathrm{Re}\left[\sum_{k=1}^{K} I_\lambda^{(k,i)}(t,\tau_k,\phi_k,\mu_{-1}^{(k)},\mu_0^{(k)})\right] \qquad \lambda=0,1, \tag{9}$$

where $I_\lambda^{(k,1)}(.,.,.,.,.)$, $k\neq 1$ is the interference at the desired receiver (referenced by $i=1$) from the *k-th* transmitter, and is given by

$$I_\lambda^{(k,1)}(t,\tau,\phi,\mu,\nu) = T^{-1}B_\lambda^{(k,1)}(t,\tau,\mu,\nu)\exp(j(\omega_0 t+\phi))\ , \tag{10}$$

where

$$B_\lambda^{(k,1)}(t,\tau,\mu,\nu) = R_{\mu,\lambda}^{(k,1)}(t,\tau)\ +\ \hat{R}_{\nu,\lambda}^{(k,1)}(t,\tau) \qquad \mu,\nu,\lambda\in\{0,1\}\ . \tag{11}$$

The functions $R_{\mu,\lambda}^{(k,m)}(t,\tau)$ and $\hat{R}_{\nu,\lambda}^{(k,m)}(t,\tau)$ are the continuous-time partial cross-correlation functions of the *k-th* and the *m-th* spectral spreading waveforms defined by

$$\begin{aligned} R_{\mu,\lambda}^{(k,m)}(t,\tau) &= (1-\mu)\int_{t-T}^{t-T+\tau} V_0^{(k)}(u-\tau+T)V_\lambda^{(m)}(u-t+T)\,du \\ &\quad +\mu\int_{t-T}^{t-T+\tau} V_1^{(k)}(u-\tau+T)V_\lambda^{(m)}(u-t+T)\,du \qquad \mu=0,1, \end{aligned} \tag{12}$$

and

$$\begin{aligned} \hat{R}_{\nu,\lambda}^{(k,m)}(t,\tau) &= (1-\nu)\int_{t-T+\tau}^{t} V_0^{(k)}(u+T-\tau)V_\lambda^{(m)}(u-t+T)\,du \\ &\quad +\nu\int_{t-T+\tau}^{t} V_1^{(k)}(u-\tau+T)V_\lambda^{(m)}(u-t+T)\,du \qquad \nu=0,1 \end{aligned} \tag{13}$$

for $\max\{0,t-T\}\leq\tau\leq\min\{T,t\}$ and $\lambda\in\{0,1\}$. Following the analytical approach introduced in [4], (11) can be shown to be

$$\begin{aligned} B_\lambda^{(k,1)}(T,\tau_k,\mu,\nu) = &\sum_{j=0}^{\gamma_k-1} V_{\mu,j-\gamma_k+N}^{(k)}[V_{\lambda,j}^{(1)}\hat{R}_\psi(S_k)+V_{\lambda,j+1}^{(1)}R_\psi(S_k)] \\ &+\sum_{j=\gamma_k}^{N-2} V_{\nu,j-\gamma_k}^{(k)}[V_{\lambda,j}^{(1)}\hat{R}_\psi(S_k)+V_{\lambda,j+1}^{(1)}R_\psi(S_k)] \\ &+V_{\nu,N-\gamma_k-1}^{(k)}V_{\lambda,N-1}^{(1)}\hat{R}_\psi(S_k)+V_{\mu,N-\gamma_k-1}^{(k)}V_{\lambda,0}^{(1)}R_\psi(S_k), \end{aligned} \tag{14}$$

where $R_\psi(\ .\ )$ and $\hat{R}_\psi(\ .\ )$ are the continuous-time partial cross-correlation functions of the chip waveform, $S_k=\tau_k-\gamma_k T_c$, and $\gamma_k=\left\lfloor\tau_k/T_c\right\rfloor$. Since τ_k is a random variable that is uniformly distributed over $[0,T)$, S_k is uniformly distributed over $[0,T_c)$, and γ_k is a random variable which is uniformly distributed over the set $\{0,1,2,\ldots\ldots,N\text{-}1\}$. The random variables γ_k and S_k are mutually independent.

For notational convenience, we define $f(S)=\hat{R}_\psi(S)+R_\psi(S)$ and $g(S)=\hat{R}_\psi(S)-R_\psi(S)$. Also, we define the set A to be the set of all nonnegative integers i less than N-1 such that $V_{\lambda,j}^{(1)}V_{\lambda,j+1}^{(1)}=1$, and the set B to be the set of all nonnegative integers i less than N-1 such that $V_{\lambda,j}^{(1)}V_{\lambda,j+1}^{(1)}=-1$. Proceeding as in [4], and given that $\gamma_k=\hat{\gamma}_k$ and $V_\lambda^{(1)}=\hat{V}_\lambda^{(1)}$, then (14) can be expressed as

$$B_\lambda^{(k,1)}(T,\tau_k,\mu,\nu)=\sum_{j\in A}U_j^{(k)}f(S_k)+\sum_{j\in B}U_j^{(k)}g(S_k)+U_{N-1}^{(k)}\hat{R}_\psi(S_k)+U_N^{(k)}R_\psi(S_k), \quad (15)$$

where the set $\{\,U_j^{(k)}\,\}$ for $0\le j\le N$, is defined by

$$U_j^{(k)}=\begin{cases} V_{\mu,j-\hat{\gamma}_k+N}^{(k)}\hat{V}_{\lambda,j}^{(1)} & j=0,1,\ldots\ldots\hat{\gamma}_k-1 \\ V_{\nu,j-\hat{\gamma}_k}^{(k)}\hat{V}_{\lambda,j}^{(1)} & j=\hat{\gamma}_k,\ldots\ldots\ldots N-2 \\ V_{\nu,N-\hat{\gamma}_k-1}^{(k)}\hat{V}_{\lambda,N-1}^{(1)} & j=N-1 \\ V_{\mu,N-\hat{\gamma}_k-1}^{(k)}\hat{V}_{\lambda,0}^{(1)} & j=N. \end{cases} \quad (16)$$

Equation (15) then reduces to

$$B_\lambda^{(k,1)}(T,\tau_k,\mu,\nu)=X_k f(S_k)+Y_k g(S_k)+P_k\hat{R}_\psi(S_k)+Q_k R_\psi(S_k)\,, \quad (17)$$

where

$$X_k=\sum_{j\in A}U_j^{(k)},\quad Y_k=\sum_{j\in B}U_j^{(k)},\quad P_k=U_{N-1},\ \text{and}\ Q_k=U_N.$$

Assuming zero is sent and using (9), (10), and (17), we can express the correlator output in the simplified form at t=T as

$$Z_\lambda^{(1)}(T)=\sqrt{P_s/2}\,\mathrm{Re}\left\{[T(1-\lambda)+\frac{\epsilon T}{N}\lambda+\sum_{k=2}^{K}B_\lambda^{(k,1)}(T,\tau_k,\mu,\nu)\exp(j\phi_k)]\exp(j\omega_0 T)\right\}. \quad (18)$$

where ϵ accounts for the non-orthogonality of the random signature sequences. With high probability, $\epsilon << N$. The probability that any two sequences will be chosen to be the same is 2^{-N}. This implies that as the sequence becomes longer ϵ/N grows smaller. In practice, the sequences for each transmitter are carefully chosen to satisfy $\epsilon << N$. Assume that $P_s=2$ and $T_c=1$; also, to simplify notation, we define $B_\lambda^{(k)}=B_\lambda^{(k,1)}(.,.,.,.)$. The sampled output of the envelope detector denoted by $\Gamma_\lambda^{(1)}=|Z_\lambda^{(1)}(T)|^2$ can now be expressed as (ignoring the ϵ^2 term)

$$\begin{aligned}\Gamma_\lambda^{(1)}=&[N^2+2N\sum_{k=2}^{K}B_\lambda^{(k)}\cos\phi_k\,](1-\lambda)+\frac{2\epsilon T}{N}\lambda\sum_{k=2}^{K}B_\lambda^{(k)}\cos\phi_k\\&+\sum_{k=2}^{K}\sum_{\ell=2}^{K}B_\lambda^{(k)}B_\lambda^{(\ell)}\cos(\phi_k-\phi_\ell),\qquad \lambda=0,1.\end{aligned} \quad (19)$$

The corresponding decision statistic $\Omega^{(1)}$ given by $\Omega_K^{(1)}=\Gamma_0^{(1)}-\Gamma_1^{(1)}$ then becomes

$$\Omega^{(1)}=\sum_{k=2}^{K}\sum_{\ell=2}^{K}[B_0^{(k)}B_0^{(\ell)}-B_1^{(k)}B_1^{(\ell)}]\cos(\phi_k-\phi_\ell)+2\sum_{k=2}^{K}(NB_0^{(k)}-\frac{\epsilon T}{N}B_1^{(k)})\cos\phi_k+N^2. \quad (20)$$

Notice that a similar set of equations is obtained if we assume one is sent.

III. CHARACTERIZATION OF THE DECISION STATISTIC

A. Quadratic Representation of Decision Statistic.

The decision statistic $\Omega_K^{(1)}$, without loss of generality, can be normalized and expressed in terms of the MAI $B_\mu^{(k)}$ as

$$\tilde{\Omega}_K^{(1)} = \sum_{k=2}^{K} \sum_{\ell=2}^{K} (\tilde{B}_0^{(k)}\tilde{B}_0^{(\ell)} - \tilde{B}_1^{(k)}\tilde{B}_1^{(\ell)})\cos(\phi_k - \phi_\ell) + 2\sum_{k=2}^{K}(\tilde{B}_0^{(k)} - \frac{\epsilon}{N}\tilde{B}_1^{(k)})\cos\phi_k + 1\,, \tag{21}$$

where $\tilde{B}_\mu^{(k)} = B_\mu^{(k)}/N$. This can be expressed in the quadratic form

$$\tilde{\Omega}_K^{(1)} = X'AX + 2B'X + c\,, \tag{22}$$

where $\mathbf{A}$ is a 2(K-1) by 2(K-1) block diagonal real symmetric matrix given by

$$\mathbf{A} = \left[\begin{array}{cccc|cccc}
1 & \cos\phi_{2,3} & \cdots & \cos\phi_{2,K} & & & & \\
\cos\phi_{2,3} & 1 & \cdots & \cdot & & & & \\
\cdot & \cdot & \cdots & \cdot & & \varnothing & & \\
\cdot & \cdot & \cdots & \cdot & & & & \\
\cdot & \cdot & \cdots & \cdot & & & & \\
\cos\phi_{2,K} & \cos\phi_{3,K} & \cdots & 1 & & & & \\
\hline
 & & & & -1 & -\cos\phi_{2,3} & \cdots & -\cos\phi_{2,K} \\
 & & & & -\cos\phi_{2,3} & -1 & \cdots & \cdot \\
 & \varnothing & & & \cdot & \cdot & \cdots & \cdot \\
 & & & & \cdot & \cdot & \cdots & \cdot \\
 & & & & \cdot & \cdot & \cdots & \cdot \\
 & & & & -\cos\phi_{2,K} & -\cos\phi_{3,K} & \cdots & -1
\end{array}\right], \tag{23}$$

with $\phi_{k,\ell} = (\phi_k - \phi_\ell)$ and $[\;]'$ denotes transposition. X, B, are both 2(K-1) dimensional vectors, defined as

$$X = [\,\tilde{B}_0^{(2)}, \tilde{B}_0^{(3)}, \ldots\ldots, \tilde{B}_0^{(K)}, \tilde{B}_1^{(2)}, \ldots\ldots\ldots, \tilde{B}_1^{(K)}]' \tag{24}$$

and

$$B = [\,\cos\phi_2, \ldots\ldots\ldots\ldots, \cos\phi_K, -\frac{\epsilon}{N}\cos\phi_2, -\frac{\epsilon}{N}\cos\phi_3, \ldots\ldots\ldots, -\frac{\epsilon}{N}\cos\phi_K]', \tag{25}$$

and $c = 1$ a scalar.

B. Gaussian Approximation for MAI

The difference in the analysis of coherent and noncoherent receivers in DS/SSMA systems arises after the correlator stage. Hence, characterizations of the decision statistic at the output of a correlator in a DS/SSMA system can apply to the analysis of a noncoherent system.

The random variables P_k, Q_k, X_k, and Y_k are conditionally independent given the signature sequences of the reference user. Under this condition, the random variables $B_\lambda^{(k,1)}(.,.,.,.)$, for $2 \leq k \leq K$, form a set of mutually independent non-Gaussian random variables. However, the normalized version of the correlator output $Z_\lambda^{(1)}(T)$ can be closely approximated by a Gaussian random variable, by using the Central Limit Theorem (CLT) for large but finite N, *provided* relative delays S_k and phases Φ_k are fixed, and for a particular set of signature sequences $\{V_0^{(1)},V_1^{(1)}\}$. This result was described as the Improved Gaussian Approximation (IGA) for MAI in [5] and is found to be accurate for any value of K. Notwithstanding, we wish to consider the conditional distribution of the MAI when only the phases are given (i.e. the reference user's sequences and the delays of the interfering signals are still random).

Note that the MAI terms $B_\mu^{(k)}$ for k=1,...,K are zero-mean with variance 2N/3 (i.e., $\tilde{B}_\mu^{(k)} \sim N(0,\frac{2}{3N})$). A comparison of the densities of $\tilde{B}_\mu^{(k)}$ and the corresponding Gaussian densities with the same variances are shown in Figure 2 for N=31 and N=127. The density of $\tilde{B}_\mu^{(k)}$ consists of a piecewise constant component and a set of impulses [4]. The impulses are represented as sufficiently narrow rectangles with the same areas as the weights of the corresponding impulses to yield Figure 2. This is justified because $B_\mu^{(k)}$ is normalized by N to yield $\tilde{B}_\mu^{(k)}$, and for large N the fixed width of a rectangle used to model an impulse in the density of $B_\mu^{(k)}$ approaches zero after normalization. The results shown in Figure 2 allow us to model the normalized decision statistic as a general quadratic form of Gaussian random variables when given only the phases ϕ_k, $2 \le k \le K$, of the interfering users.

C. The Distribution of $X'\mathbf{A}X$

For any real symmetric matrix M of order q, we can write $M=R\Lambda_q R'$, where $\Lambda_q = \mathrm{diag}(\lambda_1,\ldots\ldots,\lambda_q)$ is a diagonal matrix of the eigenvalues of M. The q×q matrix R consists of the orthonormal eigenvectors of M, such that the i-th column of R is equal to an eigenvector corresponding to λ_i, i=1,2....,q. The notation $X \sim N_q(\eta,\Sigma)$ indicates that X has a q-variate normal distribution with expectation vector η and covariance matrix Σ. X' denotes the transpose of X. The notation $X \cong Y$ means that X and Y have the same distribution, and the notation $X \equiv Y$ means that X and Y are the same random variables.

Theorem

Let $X \sim N_q(\eta,\Sigma)$, $\mathbf{A}$ be a real symmetric matrix of order q, $r=\mathrm{rank}(\Sigma\mathbf{A}\Sigma)=q$ and $\lambda_1,\lambda_2,\ldots\ldots,\lambda_r$ be the non-zero eigenvalues of $\mathbf{A}\Sigma$. Also, let S be any q×s square-root of Σ, such that $\Sigma=SS'$. If $M = S'\mathbf{A}S = R\Lambda_q R'$, then there exist real numbers $\omega_1,\omega_2,\cdots,\omega_r,\xi$ such that the inhomogeneous quadratic form $X'\mathbf{A}X + 2B'X + c$ with $B \in \mathbb{R}^q$, and $c \in \mathbb{R}$ has the distribution given by

$$X'\mathbf{A}X + 2B'X + c \cong \sum_{i=1}^{r}\lambda_i(G_i + \omega_i)^2 + \xi, \tag{26}$$

where, $G_1,G_2,\ldots.,G_r$ are independent random variables, each having a standard normal distribution and $\vec{\omega}' = (\omega_1,\ldots.,\omega_r)$ and ξ are given by

$$\vec{\omega} = \Lambda^{-1}R'S'(\mathbf{A}\eta + B) \quad \text{and} \quad \xi = \eta'\mathbf{A}\eta + 2B'\eta + c - \vec{\omega}'\Lambda\vec{\omega}. \tag{27}$$

Proof: See [6].

From the foregoing, finding the distribution of the decision statistic given knowledge of the phase references for the multiple-access interference reduces to finding the parameters $\vec{\omega}$ and ξ defined in (27).

D. Probability Density Function of $\Omega_K^{(1)}$*: Examples*

In this section, we illustrate the use of the simplified expression for the decision statistic $\Omega_K^{(1)}$. For example, the case of two transmitters (i.e., K=2), with signature sequences of length N, and a rectangular chip waveform [i.e., $\psi(t) = p_{T_c}(t)$] is chosen. The case of large K is also considered. We denote the normalized decision statistic $\tilde{\Omega}_{K}^{(1)}$ given the phase references for K users by $\hat{\Omega}_{K}$. An n-dimensional identity matrix is denoted as I_n, and $f_X(x)$ is used to represent the probability density function for a random variable X. Let $\tilde{B}_{\mu} \equiv \tilde{B}_{\mu}^{(2)}$, for μ =0,1. Note that $\cos(\phi_k - \phi_\ell) \equiv \cos\phi_k$ when the phase delays are considered modulo 2π.

a) Case 1: K=2

$$\begin{aligned} \hat{\Omega}_2 &\cong \tilde{B}_0^2 - \tilde{B}_1^2 + 2\cos\phi_2\tilde{B}_0 + 1 - \frac{2\epsilon}{N}\cos\phi_2\tilde{B}_1 \\ &\simeq \tilde{B}_0^2 - \tilde{B}_1^2 + 2\cos\phi_2\tilde{B}_0 + 1\ , \end{aligned} \tag{28}$$

where the contribution of the term multiplied by ϵ is neglected. Equation (28) can then be expressed as

$$\hat{\Omega}_2 = X'AX + 2B'X + 1\ , \tag{29}$$

where **A**, X and B are as defined in section III-B. Applying the Theorem,

$$\hat{\Omega}_2 \cong \lambda_1(G_1 + \omega_1)^2 + \lambda_2(G_2 + \omega_2)^2 + \xi, \tag{30}$$

with $\lambda_1 = -\lambda_2 = \lambda = \frac{2}{3N}$, $\omega_1 = \sqrt{3N/2}\cos\phi_2$, $\omega_2=0$, and $\xi=\sin^2\phi_2$. It can be shown that the probability density function (p.d.f) of $\hat{\Omega}_2$ is given by

$$f_{\hat{\Omega}_2}(\hat{\omega}) = \frac{1}{2\pi\lambda}\exp\left[\frac{-1}{2}\left(\frac{\hat{\omega}-\xi}{\lambda} + \omega_1^2\right)\right] \times$$

$$\int_{\Omega(\hat{\omega})} \frac{\exp(-y)}{\sqrt{y\left(\frac{\hat{\omega}-\xi}{\lambda} + y\right)}}\cosh\left[\omega_1\sqrt{\left(\frac{\hat{\omega}-\xi}{\lambda} + y\right)}\right]dy\ , \tag{31}$$

where

$$\Omega(\hat{\omega}) = \begin{cases} \left(\frac{\xi-\hat{\omega}}{\lambda}, \infty\right), & \hat{\omega} \leq \xi, \\ (0,\infty), & \hat{\omega} > \xi. \end{cases} \tag{32}$$

The p.d.f for Ω_2 is then obtained by averaging (31) over the random variable Φ. This is numerically evaluated for N=7,31,63 and plotted in Figure 3.

b) Case 2: K = ∞

After normalizing by N and setting P_s=2 and T_c=1, (18) can be written as

$$Z^{(1)}(T) = \mathrm{Re}\{[(1-\lambda) + \frac{\epsilon}{N}\lambda]\exp[j(\phi_1+\omega_0 T)] + \tilde{\mathcal{N}}_{1,\lambda} + j\tilde{\mathcal{N}}_{2,\lambda}]\} \tag{33}$$

for λ=0,1, where

$$\tilde{\mathcal{N}}_{1,\nu} = \frac{1}{N}\sum_{k=2}^{K} B_{\lambda}^{(k,1)}(T,\tau_k,\mu,\nu)\cos(\phi_k+\omega_0 T) \tag{34}$$

and

$$\tilde{\mathcal{N}}_{2,\nu} = \frac{1}{N}\sum_{k=2}^{K} B_{\lambda}^{(k,1)}(T,\tau_k,\mu,\nu)\sin(\phi_k+\omega_0 T). \tag{35}$$

The variances of $\mathcal{N}_{1,0}$, $\mathcal{N}_{1,1}$, $\mathcal{N}_{2,0}$ and $\mathcal{N}_{2,1}$ are all equal to $\frac{(K-1)}{3N}$.

The quantity $\frac{K-1}{3N}$ grows with K. Therefore, in order to study asymptotic results, we define $N_s = \frac{2N}{K-1}$ and consider the case where both N and K grow together, so that N_s remains constant. As K grows, the random variables $\mathcal{N}_{1,0}$, $\mathcal{N}_{1,1}$, $\mathcal{N}_{2,0}$ and $\mathcal{N}_{2,1}$ converge in distribution to uncorrelated zero-mean Gaussian random variables. Hence, we can model the decision statistic as a quadratic form of Gaussian random variables.

From (33), the decision statistic can be written as

$$\hat{\Omega}_\infty \cong 1 + 2(\tilde{\mathcal{N}}_{1,0}\cos\hat{\theta} + \tilde{\mathcal{N}}_{2,0}\sin\hat{\theta}) + \tilde{\mathcal{N}}_{1,0}^2 + \tilde{\mathcal{N}}_{2,0}^2 - \tilde{\mathcal{N}}_{1,1}^2 - \tilde{\mathcal{N}}_{2,1}^2, \quad (36)$$

where $\hat{\theta} = \phi_1 + \omega_0 T$. Again, the values of **A**, X and B can be extracted from (36) and the Theorem applied to obtain

$$\hat{\Omega}_\infty = \sum_{i=1}^{4} \lambda_i (U_i + \omega_i)^2 + \xi, \quad (37)$$

where $\lambda_1 = \lambda_2 = \lambda = \frac{2}{3N_s}$, $\lambda_3 = \lambda_4 = -\lambda$, and $\xi = 0$. Also, $\omega_1 = \sqrt{3N_s/2}\cos\theta$, $\omega_2 = \sqrt{3N_s/2}\sin\theta$, and $\omega_3 = \omega_4 = 0$. Thus, the p.d.f of the decision statistic can be shown to be

$$f_{\hat{\Omega}_\infty}(\hat{\omega}) = \frac{1}{4\lambda^2} \int_{\Omega(\hat{\omega})} \exp\left[\frac{-(\hat{\omega}+2w+1)}{2\lambda}\right] I_0\left(\frac{\sqrt{\hat{\omega}+w}}{\lambda}\right) dw, \quad (38)$$

where $I_0(.)$ is the zeroth-order Bessel function and

$$\Omega(\hat{\omega}) = \begin{cases} (0, \infty), & \hat{\omega} > 0, \\ (-\hat{\omega}, \infty), & \hat{\omega} \le 0. \end{cases} \quad (39)$$

It should be noted, that $f_{\hat{\Omega}_\infty}(\hat{\omega})$ is independent of the phase angle θ. Hence, $f_{\Omega_\infty}(\omega) = f_{\hat{\Omega}_\infty}(\omega)$. The numerical realization of (38) is plotted in Figure 4 for N_s=7,31,63.

IV. SUMMARY

We have obtained an expression for the decision statistic of a noncoherent receiver in a DS/SSMA communication system that employs random signature sequences. In the model, each user is assigned a pair of orthogonal, equal-energy spectral-spreading signals that are constructed from a pair of randomly generated sequences. The receiver consists of a matched filter and an envelope detector. We have proposed a quadratic Gaussian model for the multiple-access interference. In the analysis, the MAI components are assumed to be Gaussian when given the phases of all interferers, and a quadratic form of Gaussian random variables is examined. With this model, the analytical task is solving for eigenvalues and eigenfunctions to obtain the key characterizing parameters of the quadratic Gaussian model. We have shown that the decision statistic is not a non-central χ^2-distributed random variable. These results are illustrated in Figures 3 and 4.

The quadratic Gaussian model is advantageous because it provides us with a simple analytical representation for characterizing the effects of any number of users on the channel. Also, the fact that the random variables involved are independent and normal makes analysis easier and more attractive. It should be noted, that obtaining the distribution for the model is in general non-trivial. Nevertheless, we have succeeded in several cases. For these cases, the computation needed for numerical realization of the resulting distribution is quite moderate.

REFERENCES

[1] Evaggelos, A. Geraniotis, "Performance of noncoherent direct-sequence spread-spectrum multiple-access communications," *IEEE Journal Sel. Areas Commun.,* vol SAC-3, No. 5, pp 687-694, September 1985.

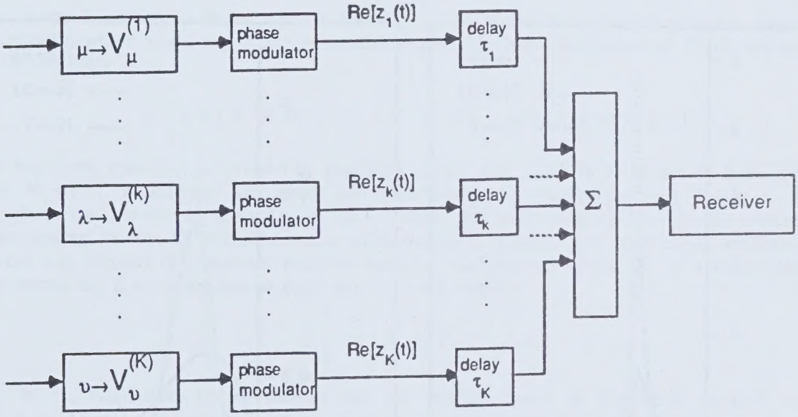
[2] George L. Turin, "The effects of multipath and fading on the performance of direct-sequence CDMA systems," *IEEE Journal Sel. Areas Commun.,* vol SAC-2, No. 4, pp 597-603, July 1984.

[3] Haige Xiang, "Binary code-division multiple-access systems operating in multipath fading, noisy channels," *IEEE Trans. Commun,* vol COM-33, No. 8, pp 775-784, Aug 1985.

[4] J.S Lehnert and M.B Pursley, "Error probabilities for binary direct- sequence spread-spectrum communications with random sequences," *IEEE Trans. Commun.,* vol. COM-35, pp. 87-98, Jan. 1987.

[5] R.K Morrow, *Bit-to-bit error dependence in direct-sequence spread-spectrum multiple-access packet radio system,* Ph.D. dissertation, Purdue Univ. West Lafayette, IN, 1988.

[6] J.J Dik and M.C.M. de Gunst, "The distribution of general quadratic forms in normal variables," *Statistica Neerlandica,* vol. 39, pp. 15-26, (1985).



$\mu \to V_\mu^{(1)}$
phase modulator
$Re[z_1(t)]$
delay τ_1
$\lambda \to V_\lambda^{(k)}$
phase modulator
$Re[z_k(t)]$
delay τ_k
$\upsilon \to V_\upsilon^{(K)}$
phase modulator
$Re[z_K(t)]$
delay τ_K
Σ
Receiver

Figure 1. A System for DS/SSMA Communication with noncoherent detection.

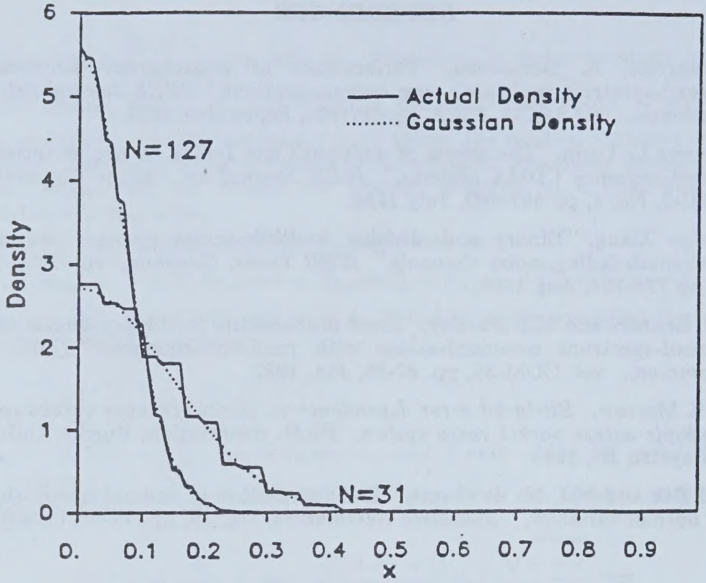


Figure 2. The density of $B_{\mu}^{(k)}$ compared with the Gaussian density with the same variance.

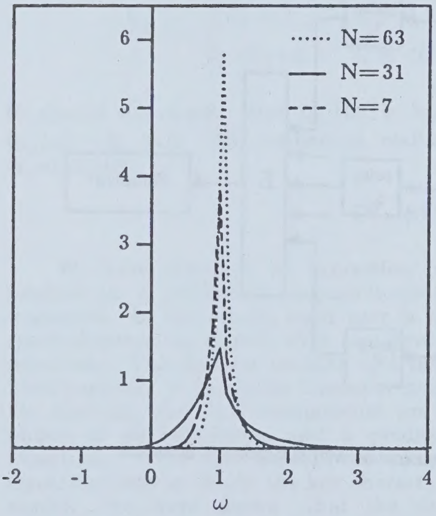


Figure 3. The density function of Ω_2.

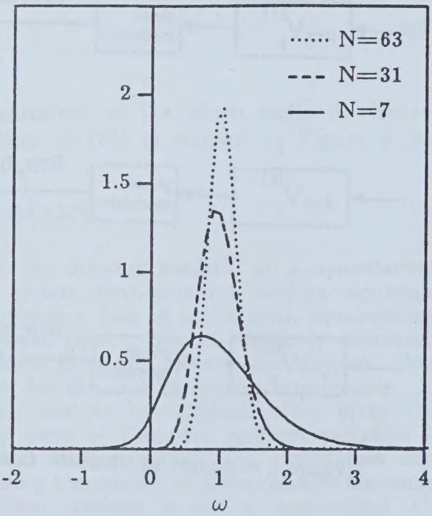


Figure 4. The density function of Ω_∞.

ON THE OPTIMALITY OF DIRECT-SEQUENCE FOR ARBITRARY INTERFERENCE REJECTION

MURAD HIZLAN and BRIAN HUGHES
Department of Electrical and Computer Engineering
The Johns Hopkins University
Baltimore, Maryland 21218

I. INTRODUCTION

Direct-sequence modulation and other spread-spectrum techniques have long been used for interference suppression in the absence of a complete statistical description of the interfering signals. Although it is well-known that biphase direct-sequence modulation with minimum distance detection is adequate in additive white Gaussian noise channels, it is not unquestionable whether other binary modulation techniques would prove to be considerably better for rejection of arbitrary interference.

Effectively, uncoded biphase direct-sequence can be viewed as one member of a broad class of robust binary communication techniques in which a data bit is transmitted by sending one out of two random processes, say $x_{-1}(t)$ and $x_{+1}(t)$, and where it is assumed that the sample paths of these processes are known to the receiver. It is of considerable interest to know whether some other form of binary signaling within this class yields significantly better performance than direct-sequence.

In this talk, we consider binary communication over a waveform channel corrupted by additive white Gaussian noise and by an unknown and arbitrary interfering signal of bounded power. We derive upper and lower bounds to the worst-case error probability of uncoded direct-sequence signaling on this channel. Furthermore, following the development in [1], we derive a lower bound to the worst-case error probability of any binary signaling technique on this channel. This bound provides a benchmark against which the performance of direct-sequence can be compared.

II. PERFORMANCE BOUNDS

Every T seconds, a binary message $b \in \{+1, -1\}$ is encoded into a signal $x_b(t)$ and transmitted to the receiver, where $x_{-1}(t)$ and $x_{+1}(t)$ are random signals which are bandlimited to W Hz and satisfy the power constraint

$$\frac{1}{T}\int_0^T x_i^2(t)\,dt \le P_T\,, \qquad i = -1, +1 \qquad \text{almost surely (a.s.)}\,. \tag{2.1}$$

The bandwidth constraint is enforced by restricting $x_{-1}(t)$ and $x_{+1}(t)$ to lie within the linear span of at most $N \doteq 2WT$ orthonormal time-limited and approximately bandlimited functions $\{\phi_1(t), \ldots, \phi_N(t)\}$. The channel is corrupted by thermal noise $n(t)$, modeled as a white Gaussian noise process with one-sided power spectral density N_0 Watts/Hz, and an unknown, arbitrary signal $s(t)$, representing interference from sources with partially or completely unknown statistics. The arbitrary signal $s(t)$ is not restricted in any way, except that it is independent of $x_b(t)$ and $n(t)$, and satisfies

$$\frac{1}{T}\int_0^T s^2(t)\,dt \le P_J \qquad \text{a.s.} \tag{2.2}$$

In [1, Proposition 1], we showed that any receiver based on the actual received waveform $y(t) = x_b(t)+n(t)+s(t)$ can, with no loss of worst-case performance, be replaced by a receiver based only on the projection of $y(t)$ onto $\{\phi_1(t), \ldots, \phi_N(t)\}$. Hence, we adopt an equivalent vector representation of the system described above, where the decisions are based upon the projection vector $\mathbf{y} = \mathbf{x}_b + \mathbf{n} + \mathbf{s}$. The power constraints on $x_b(t)$ and $s(t)$, in terms of their projections $x_{b,i}$, s_i, $1 \le i \le N$, become

$$\sum_{i=1}^{N} x_{b,i}^2 \le NE\,, \qquad \sum_{i=1}^{N} s_i^2 \le Nc \qquad \text{a.s.,} \tag{2.3}$$

where $E \equiv P_T T/N$ and $c \equiv P_J T/N$.

For any given set of signals $x_{-1}(t)$ and $x_{+1}(t)$, the worst-case error probability is defined by $\varepsilon \equiv \sup_{\mathbf{s}} \Pr\{\hat{b} \ne b\}$, where the supremum is over all $\mathbf{s}$ subject to the fundamental power limitation (2.3), and $\hat{b}$ is the estimate of the transmitted message.

Our main results are as follows: For any choice of $x_{-1}(t)$ and $x_{+1}(t)$, and for any detection rule, we show that the worst-case error probability of the resulting communication system is bounded below by

$$\varepsilon \geq \frac{\Gamma(N/2+1)}{\Gamma((N+1)/2)\sqrt{\pi}} \cdot \int_{-1}^{1} Q(\sqrt{N}\,\sigma_n[1-x/\sigma_s])\,(1-x^2)^{(N-1)/2}\,dx\ , \tag{2.4}$$

where $Q(\cdot)$ is the Gaussian tail probability function, and $\sigma_n^2 \equiv 2E/N_0$, $\sigma_s^2 \equiv E/c$ are *signal-to-thermal-noise* and *signal-to-interference* ratios, respectively.

For the special case of uncoded direct-sequence modulation, $\mathbf{x}_b \equiv b\sqrt{E}\,\boldsymbol{\alpha}$ where $\boldsymbol{\alpha}$ is an N-vector of i.i.d. Bernoulli random variables representing the pseudorandom *signature sequence.* We show that the worst-case error probability of direct-sequence, denoted by ε_{ds}, is bounded by

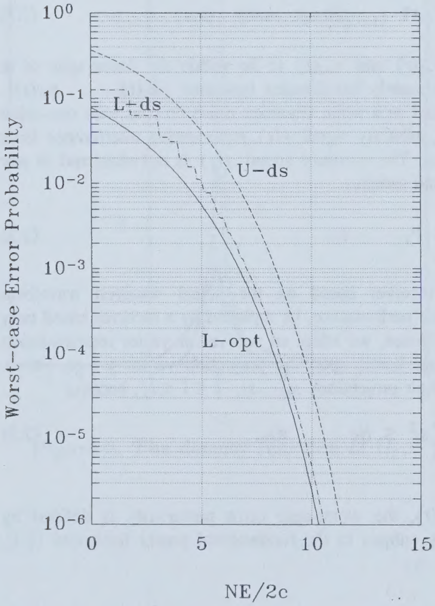
$$\exp(-N\,E(\sigma_n,\sigma_s)) \geq \varepsilon_{\text{ds}} \geq \max_{1 \leq k \leq N} 2^{-k} \sum_{i=1}^{k} \binom{k}{i} Q\left(\frac{\sigma_n}{\sigma_s}[\sigma_s\sqrt{N} + \sqrt{k} - 2i/\sqrt{k}\,]\right]\ , \tag{2.5}$$

where $E(\sigma_n,\sigma_s) \equiv \max_{\lambda \geq 0} [\lambda\sigma_s - (\lambda\sigma_s/\sigma_n)^2/2 - \ln(\cosh(\lambda))]$.

As an example, we plot these bounds in the Figure below for $N = 256$ as a function of $NE/2c$ in the absence of thermal noise ($N_0 = 0$). We observe that below the 10^{-3} level, the difference between the upper bound for direct-sequence and the lower bound for any choice of signals and detection rule is approximately 1 dB. The actual difference between the exact curves is likely to be much less, given the losses introduced by the bounds. This behavior is typical for other values of N_0 and N, as well.

We conclude that the worst-case performance of uncoded BPSK direct-sequence modulation is very close to the optimal worst-case performance achievable by any other choice of signals $x_{-1}(t)$ and $x_{+1}(t)$, and any detection rule. In particular, one cannot hope to gain any more than 1 dB in terms of worst-case error probability by switching from direct-sequence to any other binary communication technique in the general class.

As a final remark, we note that this behavior does not generalize to arbitrary, one-coded-bit-per-chip direct-sequence modulated, soft-decision decoded binary block codes. On the contrary, we have shown in [2] through a comparison of asymptotic error exponents that the worst-case performance of low-minimum-distance codes with direct-sequence is considerably far from that achievable by other forms of modulation.



REFERENCES:

[1] B. Hughes and M. Hizlan, "An asymptotically optimal random modem and detector for robust communication," Dept. of Elect. Comp. Eng., The Johns Hopkins University, Tech. Rep. 89-06, May 1989.

[2] M. Hizlan and B. Hughes, "Worst-case performance of direct-sequence modulated codes on channels with unknown interference," Dept. of Elect. Comp. Eng., The Johns Hopkins University, Tech Rep. 89-09, June 1998.

Jointly Optimal Receivers: An Application to Joint Acquisition and Detection for DS-SS Systems

COSTAS N. GEORGHIADES
Electrical Engineering Department
Texas A&M University
College Station, TX 77843

Abstract

Recent general results on the problem of joint sequence and timing estimation from samples of the output of a matched filter, taken asynchronously with the actual timing and at arbitrary sampling rates are applied to the problem of joint acquisition and detection for direct-sequence, spread-spectrum (DS-SS) systems. Lower bounds on the expected bit-error probability and mean-square synchronization error are presented.

Introduction

Recently there has been much interest in the study of algorithms that process data obtained by sampling the output of a matched filter at some low rate [1,2,3,4,5]. Most of the work has concentrated on the problem of deriving timing estimates from these samples, using mostly decision directed techniques. In [4], the problem of jointly estimating sequences and timing is investigated mainly for NRZ signaling and single samples per symbol. Subsequent work in [5] has extended these results to arbitrary pulse shapes and multiple samples per symbol. In this paper, the general results in [2] are used to obtain lower bounds on the mean-square synchronization error and error probability for receivers that jointly estimate sequences and symbol-timing in DS-SS systems.

The Joint Likelihood Function

The following joint likelihood function is derived in [2]

$$\ell(\mathbf{a},\tau)=\sum_m a_m v_m(\tau)-\frac{\sqrt{E_B}}{2}\sum_{m,n}a_m a_n M_{m-n}(\tau), \tag{1}$$

where $\{a_m\}$ is a sequence of modulation symbols taking values from some discrete alphabet, and τ is a random delay due to the asynchronous sampling of the matched-filter output. The function $v_m(\tau)$ represents the interpolated and down-sampled data obtained from the samples of the matched-filter taken at some integer sampling rate L in samples per symbol. The quantity $M_n(\tau)$ in (1), which is directly related to the error-probability is

$$M_n(\tau)=\int_{-1/2T_s}^{1/2T_s}\frac{\left|\sum_k R(f-k/T_s)\exp(-j2\pi k\tau/T_s)\right|^2}{\sum_m R(f-m/T_s)}\cos(2\pi f nT)df, \tag{2}$$

where $T_s=T/L$ is the sampling period, $1/T$ is the symbol rate, and $R(f)$ is the Fourier transform of the autocorrelation function $r(t)$ given by (for $p(t)$ some arbitrary baseband pulse-shape) $r(t)=\int_{-\infty}^{\infty}p(t+u)p(u)du$.

The following lower-bounds are derived in [5]

$$E[(\hat{\tau}-\tau)^2]\geq\left\{(2E_B/N_0)N\int_{-\infty}^{\infty}4\pi^2f^2R(f)\frac{R(f)}{\sum_m R(f-m/T_s)}df\right\}^{-1} \tag{3}$$

$$P(e|\tau) \geq \frac{1}{2}\text{erfc}\left[\sqrt{\frac{E_B}{N_0}M_0(\tau)}\right]. \qquad (4)$$

Performance Bounds for DS-SS Systems

Under direct-sequence spread-spectrum, each positive or negative rectangular pulse representing a symbol is multiplied by a spreading pseudo-random sequence of some length M. If we let the sequence be denoted by $\{p_n\}$, then the spreading signal is $p(t) = \sum_{n=0}^{M-1} p_n\phi(t - nT_c)$ where $\phi(t)$ is a rectangular pulse of duration $T_c = T/M$ seconds. Then the autocorrelation function of the spread signal is $r(t) = \sum_{n=0}^{M-1}\sum_{m=0}^{M-1} p_m p_n c[t+(n-m)T_c]$ where $c(t)$ is the autocorrelation function of $\phi(t)$ which has a Fourier transform $C(f) = (T_c/M)[\sin(\pi f T_c)/(\pi f T_c)]^2$. It follows $R(f) = C(f)\left|\sum_{m=0}^{M-1} p_m e^{-j2\pi f m T_c}\right|^2$. Let the sampling be at the rate of Q samples per chip, ($L = QM$ samples per symbol). Then (3) becomes

$$E[(\frac{\hat{\tau}-\tau}{T})^2] \geq \left\{(2E_B/N_0)8NL\int_0^{1/2}\sin^2(\pi Qx)\left|\sum_{m=0}^{M-1} p_m e^{-j2\pi mQx}\right|^2 dx\right\}^{-1}. \qquad (5)$$

Also, $M_0(\tau)$ in (4) becomes

$$M_0(\tau) = \frac{2}{L}\int_0^{1/2}\frac{\sin^2(\pi Qx)}{\sin^2(\pi x)}[1 - 4\frac{|\tau|}{T_s}(1-\frac{|\tau|}{T_s})\sin^2(\pi x)]\left|\sum_{m=0}^{M-1} p_m e^{-j2\pi mQx}\right|^2 dx. \qquad (6)$$

From (6) it can be seen that worst-case $M_0(\tau)$ is at $\tau = T_s/2$. For the length 15 maximal-length sequence 1-1-1-11111-11-111-1-1, this corresponds to $M_0(T_s/2) = 1/2$, i.e., 3dB below perfect synchronization performance at one sample per chip ($Q = 1$), $M_0(T_s/2) = 3/4$ (1.25 dB loss) for $Q = 2$ and $M_0(T_s/2) = 7/8$ (0.6 dB loss) for $Q = 4$. For the same sequence, the right-hand side of (3) becomes $1/(E_B/N_0)30NL$. We note that an m-sequence may not necessarily be the best choice for a joint synchronization and detection algorithm, and that in general a sequence must be chosen that yields a good compromise between a good error-probability and a good mean-square error lower-bound.

iv. References

1. K.H. Mueller and M. Muller, "Timing Recovery in Digital Synchronous Data Receivers," IEEE Transactions on Communications, Vol. COM-24, May 1976.
2. F.M. Gardner, "A BPSK/QPSK Timing-Error Detector for Sampled Receivers," IEEE Transactions on Communications, Vol. COM-33, June 1986.
3. M. Oerder and H. Meyr, "Digital Filter and Square Timing Recovery," IEEE Transactions on Communications, Vol. COM-36, May 1988.
4. C.N. Georghiades, "Optimum Delay and Sequence Estimation From Incomplete Data," to appear in the IEEE Transactions on Information Theory.
5. C.N. Georghiades and Marc Moeneclaey, "Sequence Estimation and Synchronization from Multiple Samples per Symbol," under review.

PERFORMANCE OF NONCOHERENT FREQUENCY HOPPED M-ARY SHIFT KEYING (FH-MFSK) COMMUNICATIONS OVER FACTORY RADIO CHANNEL

PARVIZ YEGANI AND CLARE D. MCGILLEM
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Abstract- The performance of a Frequency Hopped M-ary Frequency Shift Keyed Spread-Spectrum Communication System over Factory Radio Channel is considered. The average Bit Error Rate (BER) is formulated and is evaluated approximately using Stirling's Formula. The number of simultaneous users in terms of number of hopped frequencies, number of MFSK chips, receiver threshold, signal-to-noise ratio and channel statistics at a fixed BER for a Rayleigh fading channel is determined.

I. INTRODUCTION

In order to provide digital communication services to a large number of users spread-spectrum modulation techniques have been investigated by many researchers. In practice, timing and synchronization must be established to within a fraction of the chip interval. Because of this the FH technique which has a longer chip interval has the major advantage of being more robust than Direct Sequence (DS) techniques for mobile users. There are cases where the rms delay spread is small and the signal components are not resolvable for reasonable chip durations. For example, in a typical factory environment with an rms delay of about 200 ns using FH-MFSK offers promise of reducing the serious fading that occurs in the channel. The Frequency-Hopped Differential Phase Shift keying (FH-DPSK) spread spectrum multiple access communication system was first proposed by Cooper[1-3]. Cooper [1] suggested that the FH system and the existing narrowband system can use the same frequency band without any further performance degradation. Viterbi in his pioneering work initiated the use of Multiple Frequency Shift Keying (MFSK) modulation with Code Division Multiple Access (CDMA) by frequency hopping over the 2^m frequencies in the common bandwidth W over a radio channel. Error control coding was employed to minimize the effect of mutual interference among the intended users as well as the effects of noise and external users. Goodman et al. [5] extended the technique proposed by Viterbi for multiple access communications among mobile radios.

In this paper the Noncoherent Synchronous FH-MFSK system over the factory radio channel [6-11] is considered and the system performance is compared with that of other systems. As stated earlier the timing requirements in FH systems is not as stringent as in DS system. Therefore, to have a factory radio network using FH-MFSK waveforms with synchronized hopping patterns seems quite attractive. In order to evaluate such a system average probability of error for the system is computed and the number of simultaneous users for a prespecified probability of error in terms of system parameters and channel statistics is investigated. The factory channel is assumed to be Rayleigh. The performance evaluation of the system for more complicated channel models awaits further study. Noncoherent demodulation of the data is employed and a majority logic decision is used to declare the signal present.

II. FH-MFSK System Description

2.1. Transmitter

The FH-MFSK system block diagram is shown in Fig. 1. In the transmitter (Fig. 1a) data modulation (MFSK) and frequency hopping modulation (FH) have been implemented in a two-step modulation process. MFSK modulation provides the multiple-access capability to the system while the FH modulation protects the system against multipath losses. The MFSK modulator produces an M-ary signal based on the simultaneous dictates of the PN code and the data. Then each MFSK chip modulates an RF carrier produced by the frequency synthesizer.

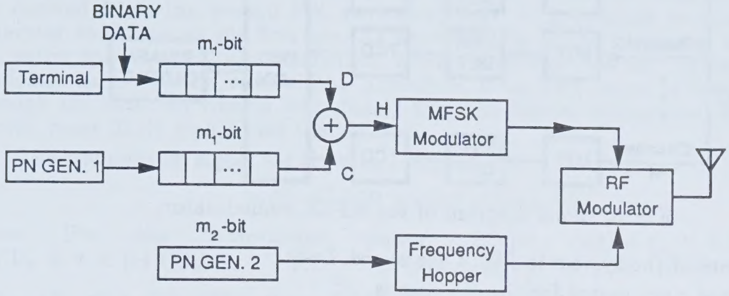
The time-frequency diagram of the FH-MFSK system is sketched in Fig. 2. The MFSK chip rate and the symbol rate are the same ($R_h = R_s$). Because of the interleaving each m_1-bit data word consists of four different data bits. In other words, during each chip interval, four different data bits (one MFSK chip) are transmitted. Chip tones corresponding to the same data bit appear in different hop intervals. That is, each data bit is transmitted four times, every time via different chip tones. Since every single tone is hopped independently, the different copies of each data bit passing through the channel experience different transmission losses such as fading, interchannel interferences, etc.

2.2. Receiver

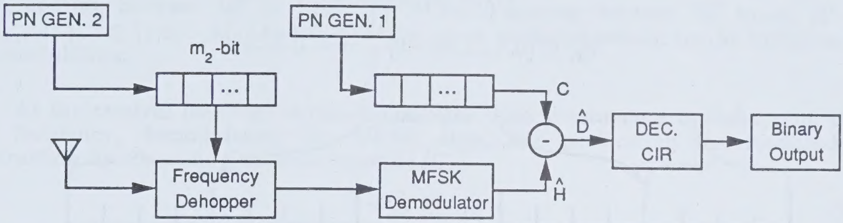
The receiver block diagram is shown in Fig. 1b. It is assumed that the PN generator of the receiver produces the same hopping pattern as the one in the transmitter. The FH demodulator dehops an MFSK chip every symbol duration. The MFSK demodulator uses a hard quantizer to detect individual data bits. At the front end of this modulator a group of Matched Tuned Filters (MTF) are used to separate chip tones from each other. Each filter is followed by an Envelop Detector (ENV DET) and a Threshold Circuit Decision (TCD). Finally, a decision Matrix (DEC MTX) is used to declare the presence of the signal. The MFSK demodulator block-diagram is shown in Fig. 3. TCD sometimes can make wrong decisions due to either multiple-access interference or multipath interference. These two cases are described by: *Miss*, a decision of no tone being present when in fact a chip tone has been transmitted at the frequency and *False Alarm*, a decision of a tone being present at a frequency may be made when none has been transmitted.

III. FH-MFSK System Analysis

An example of the construction of a data vector is shown in Fig. 4. Every T_h seconds (signaling interval) a data word (m_1-bit MFSK chip) is repeatedly transmitted L times. The duration of each chip is $T_h (= T/L)$ time units, where L is the number of times each word is repeated.


BINARY DATA
Terminal
m_1-bit
PN GEN. 1
m_1-bit
D
H
C
MFSK Modulator
RF Modulator
m_2-bit
PN GEN. 2
Frequency Hopper

(a) Transmitter [TX]


PN GEN. 2
PN GEN. 1
m_2-bit
C
$\hat{D}$
DEC. CIR
Binary Output
Frequency Dehopper
MFSK Demodulator
$\hat{H}$

(b) Receiver [RX]

Fig. 1. Block diagram of FH-MFSK system (a) Transmitter (b) Receiver

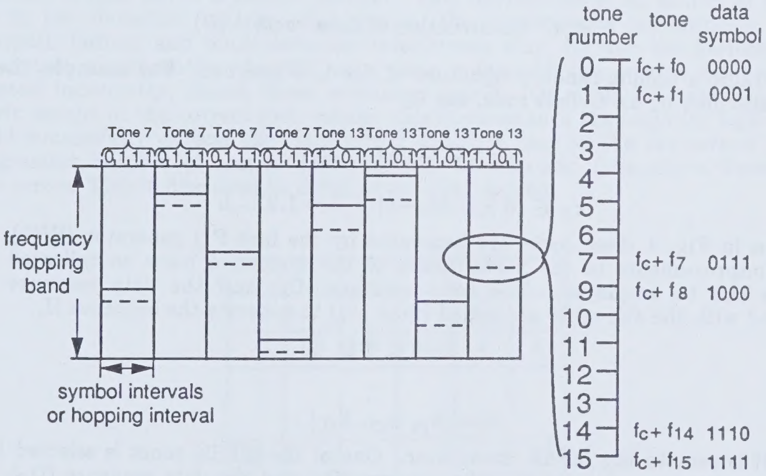

Tone 7
Tone 7
Tone 7
Tone 7
Tone 13
Tone 13
Tone 13
Tone 13
0,1,1,1
0,1,1,1
0,1,1,1
0,1,1,1
1,1,0,1
1,1,0,1
1,1,0,1
1,1,0,1
frequency hopping band
symbol intervals or hopping interval
tone number
tone
data symbol
0 $f_c + f_0$ 0000
1 $f_c + f_1$ 0001
2
3
4
5
6
7 $f_c + f_7$ 0111
9 $f_c + f_8$ 1000
10
11
12
13
14 $f_c + f_{14}$ 1110
15 $f_c + f_{15}$ 1111

Fig. 2. FH-MFSK time-frequency diagram

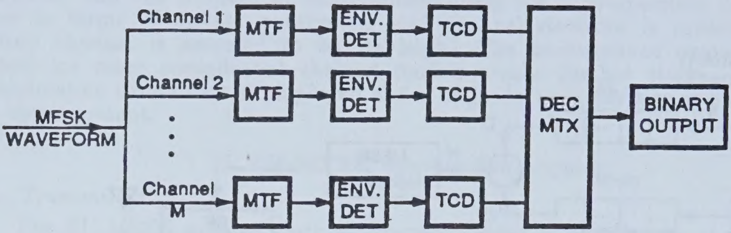


Fig. 3. Block diagram of the MFSK demodulator.

The parameters of the system in Fig. 4 are $m_1=4$, $L=5$.
The constructed data vector for the ℓth user is

$$D_\ell = [d_{\ell 1} d_{\ell 2} ... d_{\ell L}]^t \tag{1a}$$

$$d_{\ell i} \in \{0,1,...,M_1-1\}\,, \quad i = 1,2,...,L$$

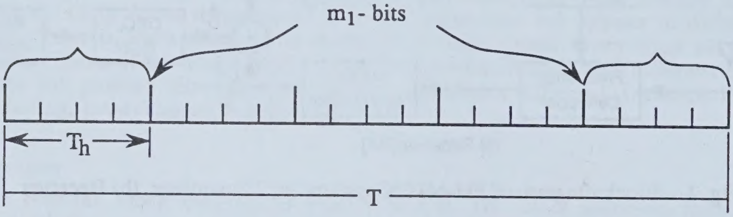


Fig. 4. Construction of data vectors (D)

To each radio a unique random signature of size L is assigned. For example, the ℓth radio is specified by an L-digit code, say C_ℓ

$$C_\ell = [c_{\ell 1} c_{\ell 2} ... c_{\ell L}]^t \tag{1b}$$

$$c_{\ell i} \in \{0,1,...,M_1-1\}\,, \quad i = 1,2,...,L$$

As shown in Fig. 4 these codes are generated by the first PN generator (PN1). For further improvements in the performance of the system, a more complicated code sequence can be employed. The code sequence (C_ℓ) and the data sequence (D_ℓ) associated with the ℓth radio are added (mod M_1) to generate the sequence H,

$$H = C \oplus D \tag{2}$$

where

$$H = [h_{\ell 1}\ h_{\ell 2} ... h_{\ell L}]\,. \tag{3}$$

Vector H is sent to the MFSK modulator. One of the MFSK tones is selected based on the simultaneous dictates of code sequence (C_ℓ) and the data sequence (D_ℓ). The process of superimposing these two sequences gives the error correction capability to the system. That is, to some extent the errors due to transmission losses (e.g. the multi-path fading,...) and the multiple access interference can be detected and corrected. The number of errors being corrected would be a function of the parameters of the code sequence such as the code length, the minimum hamming distance, the decision policy, etc.

The MFSK waveforms are used to modulate the RF signals generated by the frequency synthesizer. The hopping patterns which excite the frequency synthesizer are derived from the second PN generator(PN2). It is possible to choose this PN generator the same as the first one. As shown in Fig. 2 the frequency hopper covers the entire spread spectrum bandwidth. Every T seconds the RF carrier is switched randomly across the SS bandwidth. Therefore, if the first frame of data does not go through the channel(due to deep fade) the next frame, transmitted on a different carrier, more likely would pass the channel.

The transmitted signal for the ℓ-th radio is

$$s_\ell(t) = \sqrt{2P}\ \cos[2\pi\tilde{f}_\ell(t)t+\tilde{\phi}_\ell(t)] \tag{4}$$

where $P=$ the transmitted power, $\tilde{f}_\ell(t)= f_\ell(t) + f_m(t) + f_0$, $f_\ell(t) = f_i,\ iT_h \leq t < (i+1)T_h$, $f_i \in \{\nu_1,\ldots,\nu_{M_2}\}$, $i \in \{1,2,\ldots,M_2\}$, $f_m(t) = d_\ell\Delta$, $iT_h \leq t < (i+1)T_h$, $d_\ell = \sum_{i=1}^{m_1} d_{\ell i}\, 2^i$, $\begin{matrix} d_{\ell i} \in \{0,1\} \\ \ell \in \{1,2,\ldots,L\} \end{matrix}$, $f_0 =$ RF carrier , $\Delta \geq T_h^{-1}$, spacing between MFSK tones, $\Delta' > M_1\Delta$, spacing between RF tones, $\Delta' = |\nu_{i+1}-\nu_i|$, $i \in \{1,2,\ldots,M_2\}$ and $\tilde{\phi}_\ell(t) =$ the phase angle introduced by the MFSK and FH modulators.

At the receiver (see Fig. 1b)the transmitted data is recovered by dehopping the RF frequency, demodulating the MFSK chip, and performing the modulo-M_1 subtraction by the same signature sequence (C_ℓ).

$$\hat{D} = \hat{H} \ominus \ \hat{C} \tag{5}$$

$$\hat{H} = H\ ,\ \ \hat{D} = D \tag{6}$$

This corresponds to an ideal case where the transmission is perfect and there is no interfering signal. The detection matrix (Fig. 5) illustrates the reception of the transmitted data (X) in a perfect channel. This corresponds to all entries of a correct row in the detection matrix. In practice, the transmission impairments such as multipath fading, and multiple-access interference may degrade the performance of the system significantly. In this case some entries of the correct row would be detected incorrectly. Hence these erroneous entries (O) could be anywhere in the matrix except in the correct row. Under this circumstance the majority logic receiver would successfully detects the transmitted signal as long as the the correct row has the greatest number of entries. However, due to *miss* and *false alarm* there will be some errors. This is discussed in detail in the next section.

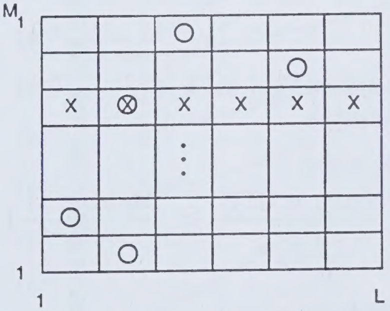


Fig. 5. Detection Matrix of the FH-MFSK System.

IV. Performance of FH-MFSK System

The bit error probability for the FH-MFSK can be written as

$$P_e = P_{e,0}(1 - \hat{P}) + P_{e,1}\hat{P} \tag{7}$$

In writing (7) it is assumed that the channel is nonselective and the the hopping patterns are stationary. $P_{e,0}$ is the conditional probability of error for any bit interval given that there are no hits and $P_{e,1}$ is the conditional probability of error given there is at least one hit. These probabilities are computed from eqn. (3-A), and the four-term *Stirling's approximation* is used to deal with the computational problems. $\hat{P}$ is the probability of a hit occurring which is fixed from bit to bit for stationary random hopping patterns. If the patterns are mutually independent and identically distributed then

$$\hat{P} = 1 - (1 - p)^{K-1} \tag{8}$$

where K is the number of simultaneous users and $p = 2^{-m_2}$. As shown in the Appendix the channel probabilities (P_D and P_F) along with the system parameters are needed in order to compute the bit error probability. For the Rayleigh fading channel these probabilities for On-Off Keying (OOK) modulation scheme are given by [13]

$$P_F = \exp(-\frac{\beta^2}{2}) \tag{9a}$$

and

$$P_D = 1 - \exp(-\frac{\beta^2}{2(1+\gamma)}) \tag{9b}$$

where β is the normalized receiver threshold. To find the optimum parameter set for the system the design criteria are determined as follows.

The RF band width which is required to account for the orthogonality of hopped frequencies is

$$\begin{aligned} W_{SS} &= \nu_{M_2} - \nu_1 + M_1\Delta, & \Delta &\geq T_h^{-1} \\ &= M_2\hat{\Delta}, & \hat{\Delta} &\geq M_1\Delta + T_h^{-1} \\ &= M_2(M_1+1)\Delta, & &\geq (M_1+1)\Delta \end{aligned} \tag{10}$$

the data rate is given by

$$R = \frac{m_1}{L\ T_h} \tag{11}$$

and the lower bound on the system band width is

$$W = M_2(M_1 + 1)T_h^{-1} \tag{12}$$

The normalized data rate (r) is then

$$r^{-1} = \frac{W}{R} \geq \frac{M_2(M_1 + 1)T_h^{-1}}{\frac{m_1}{L}T_h^{-1}} \geq \frac{M_2(M_1 + 1)}{\log M_1} L \tag{13}$$

which gives an upper limit on L

$$L \leq \frac{\log M_1}{M_2(M_1 + 1)} r^{-1} \tag{14}$$

As an example, let $W_{SS} = 256$ MHz, R = 100 Kpbs, $M_1 = 256$, and $M_2 = 16$. Then $L \leq 5$ and the fractional system spectrum efficiency is η=100KR/W=K/25.6.

V. Results

The error performance of the FH-MFSK system over a Rayleigh fading channel is shown in Fig. 6. It is seen from the figure that by choosing an appropriate signal to noise ratio (γ) at a fixed receiver threshold (β) a large number of users (K) can communicate within the system maintaining a prespecified Bit Error Rate (BER). These results indicate that with this system a higher number of users can communicate under the specific conditions than can with more conventional systems[1-5]. For example, for a threshold of 4 and SNR of 30dB with 8-ary FSK modulation potentially more than 150 user can simultaneously transmits their data with an BER 10^{-4} or better. With a perfect transmission (P_D=P_F=0) where the system suffers only from mutual interference, an error rate of less than 10^{-4} can be maintained with up to 220 simultaneous transmitting users(terminals). It was also noticed that the system is robust and the performance degradation due to the worst condition in the channel was minimized. The spectrum efficiency corresponding to a perfect transmission condition is about 9 percent. This is better than the efficiency of the previously described systems [1],[5]. The coherence bandwidth for a factory radio channel is around 5 to 10 MHz which corresponds to an rms delay spread of 100 to 200 ns [6-8]. This suggests that at least 10 MHz frequency separation between adjacent frequency channels is required so that each frequency channel will fade independently. Therefore much higher bandwidth is to be used and larger number of frequencies are to be hopped for the system to be (more) effective against multipath. In the system considered here the number of hopping frequencies was 16, the RF bandwidth 256 MHz and the data rate was 100 Kbits/sec. Each MFSK chip was repeated five times (L = 5). The spacing of 16 MHz between adjacent carriers has been selected such that each carrier fades independently as long as the rms delay spread of the multipath channel is not less than about 60 ns. Therefore the 20 MHz RF bandwidth adopted in [5] would not not be adequate to overcome the multipath in a very harsh multipath environment.

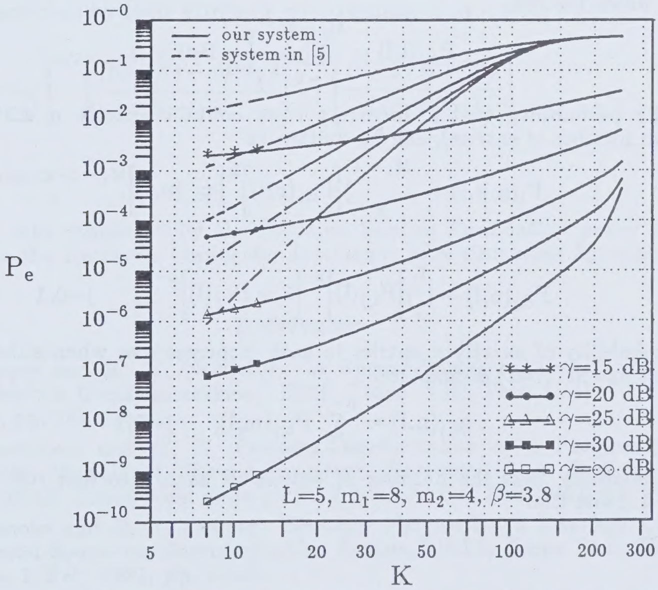


Fig. 6. Performance of FH-MFSK system over Rayleigh fading channel.

Appendix

Hit probability from eqn. (8) of section IV is given as

$$\hat{P} = 1 - (1 - \frac{1}{M_2})^{K-1} \tag{1-A}$$

and bit error probability is (eqn. (7) section IV)

$$P_e = P_{e,0}\,(1 - \hat{P}) + P_{e,1}\,\hat{P} \tag{2-A}$$

The bit error probability for M-ary orthogonal signaling is given by [12]

$$P_{e,j} = \frac{M_1/2}{M_1 - 1} P_{\omega,j}, \qquad j=0,1 \tag{3-A}$$

For the cases of no hit (j=0) or at least one hit (j=1). Then the word error probabilities are

$$P_{\omega,j} = 1 - P_{c,j}, \qquad j=0,1 \tag{4-A}$$

where $P_{c,0}$ and $P_{c,1}$ are the overall probabilities of correct decision when no hit or at least one hit occurs, respectively. These probabilities are computed as follows.

Let l be the number of frequencies (ν's) which overlap at any signaling interval (T_h). From this point on the argument l in any equation indicates that there are only l users which simultaneously transmit on the same frequency. Then the correct decision probabilities for this number of frequency channels can be written as

$$P_{c,j}(l) = \sum_{i=0}^{L} P_r(i)\, P_{c,j}(i,l) \tag{5-A}$$

where $P_r(i)$ is the probability of i entries in the correct row which is given by

$$P_r(i) = \binom{L}{i}(1 - P_D)^i P_D^{L-i} \tag{6-A}$$

and $P_{c,j}(i,l)$ is the probability of correct detection given that the correct row contains i entries which follows

$$P_{c,j}(i,l) = \sum_{k=0}^{M_1-1} \frac{1}{k+1} P_j(i,k,l) \tag{7-A}$$

where the probability that k incorrect rows contain exactly n entries (n is the maximum number of entries), can be written as

$$P_j(n,k,l) = \binom{M_1-1}{k}\left[P_{\bar{c},j}(n,l)\right]^k \left[q_{\bar{c},j}(n,l)\right]^{M_1-1-k} \tag{8-A}$$

where

$$P_{\bar{c},j}(n,l) = \binom{L}{n}\left[P_{I,j}(l)\right]^n \left[1 - P_{I,j}(l)\right]^{L-n}, \qquad j=0,1 \tag{9-A}$$

is the probability of exactly n entries in each incorrect row when either no hits (j=0) or at least one hit (j=1) occurs. and

$$q_{\bar{c},j}(n,l) = \sum_{m=o}^{n-1} P_{\bar{c},j}(m,l), \qquad j=0,1 \tag{10-A}$$

is the probability that the number of entries in each incorrect row is less than or equal n—1. Note that

$$P_{I,j}(l) = P_{INT,j}(l) + P_F - P_{INT,j}(l)\,P_F, \quad j=0,1 \tag{11-A}$$

is the probability of insertion due to either multiple access interference or noise. The probability of insertion due to multiple access interference is

$$P_{INT,j}(l) = \begin{cases} 0, & j=0 \\ P_{INT}(l)\,(1-P_D), & j=1 \end{cases} \tag{12-A}$$

where

$$P_{INT}(l) = 1 - (1 - \frac{1}{M_1})^l \tag{13-A}$$

Now we can write (for any value of l)

$$P_{c,0} = P_{c,0}(l) \tag{14-A}$$

and

$$P_{c,1} = \sum_{l=1}^{K-1} P_{c,1}(l)\,P_l \tag{15-A}$$

where

$$P_l = \binom{K-1}{l}(\frac{1}{M_2})^l(1-\frac{1}{M_2})^{K-1-l} \tag{16-A}$$

For a large number of users the term $\binom{K-1}{l}$ in eqn. (15-A) may cause computational problems. To deal with this problem we use *Stirling's Formula* to evaluate the factorial function. For large values of m the one-term Stirling approximation of m! is given by [14]

$$m! = \sqrt{(2\pi m)}\; m^m \exp(-m) \tag{17-A}$$

Replacing all factorials by their Stirling's approximations in (16-A) we get

$$P_l = \left[\frac{K-1}{2\pi l(K-1-l)}\right]^{1/2}\left[\frac{(K-1)(1-1/M_2)}{K-1-l}\right]^{K-1}\left[\frac{(K-1-l)\,1/M_2}{l(1-1/M_2)}\right]^{l} \tag{18-A}$$

Acknowledgment

This work was supported by the National Science Foundation under the grant CDR8803017 to the Research Center for Intelligent Manufacturing Systems, Purdue University.

References

[1] G. R. Cooper and R. W. Nettleton, "A Spread-Spectrum Technique for High-Capacity Mobile Communications," IEEE Trans. Veh. Tech., Vol. VT-27, No. 4, Nov. 1978, pp. 264-275.

[2] R. W. Nettleton and G. R. Cooper, "Performance of a Frequency-Hopped Differentially Modulated Spread-Spectrum Receiver in a Rayleigh Fading Channel," IEEE Trans. Veh. Tech., Vol.-VT-30, Nol. 1, Feb. 1981, pp. 14-29.

[3] M. Matsumoto and G. R. Cooper, "Multiple Narrow-Band Interferers in a FH-DPSK Spread-Spectrum Communication System," IEEE Trans. Veh. Tech., Vol. VT-30, No. 1, Feb. 1981, pp. 37-42.

[4] A. J. Viterbi, "A Processing-Satellite Transponder for Multiple Access by Low Rate Mobile Users," Proc. *Digital Satellite* Commun. Conf. Montreal, October 1978 pp. 166-174.

[5] D. J. Goodman, P. S. Henry, and V. K. Prabhu, "frequency-Hopped Multilevel FSK for Mobile Radio," B.S.T.J., Vol. 59, No. 7, Sept. 1980, pp. 1257-75.

[6] T. S. Rappaport, "Characterization of UHF Factory Multipath Channels," PhD. Dissertation, School of Electrical Engineering, Purdue University, West Lafayette, Indiana, June 1988.

[7] T. S. Rappaport, C. D. McGillem, "UHF Fading in Factories," IEEE J. Sel. Areas Comm., Vol. SAC-7, No. 1, January 1989, pp. 40-48.

[8] T. S. Rappaport, C. D. McGillem, "UHF Multipath and Propagation Measurements in Manufacturing Environments," Proc. IEEE Global Communication Conference, November 30,1988, Hollywood, FL, Session 21.5.

[9] P. Yegani and C. D. McGillem, "A Statistical Model for the Line-Of-Site (LOS) Factory Radio Channel," Proc. 39th IEEE Vehicular Technology Conference, San Francisco, May 1989, pp. 496-503.

[10] P. Yegani and C. D. McGillem, "A Statistical Model for the Obstructed (OBS) Factory Radio Channel," Proc. IEEE Global Telecommunications Conference & Exhibition , Dallas, Texas, Nov. 27-30, 1989.

[11] P. Yegani and C. D. McGillem, "A Statistical Model for Factory Radio Channel," to appear in IEEE Trans. on Commun., future issue.

[12] J. G. Proakis, Digital Communications, McGraw-Hill, Inc., 1983.

[13] Mischa Schwartz, W. R. Bennett, and St. Stein, Communication Systems and Techniques, McGraw-Hill, Inc., 1966.

[14] Thornton C. Fry, Probability and its Engineering Uses , D. Van Nostrand Company, Inc., second Ed. 1965.

A MULTITONE FH/MFSK SYSTEM ON PARTIAL BAND INTERFERENCE AND NOISE

G. E. Atkin and H. P. Corrales

Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, IL 60616

The simplest antijamming scheme consists in subdividing the signalling interval into several time slots, or chips, and transmitting the same information over randomly selected frequency bands on each one of these time slots. The increased redundancy of this Frequency-Hopping (FH) system reduces the probability of interference; however, the transmission rate must be reduced to accommodate the bandwidth expansion resultant from this subdivision [1]. In this paper we suggest an alternative technique to Time Diversity FH/MFSK. The antijamming scheme proposed here consists on selecting several frequencies on each signalling interval. The allocation of these frequencies is based on blocks of elements derived from a combinatorial construction known as *Steiner t-Design* [2]. It was proved in [3] that on a noiseless interference channel this scheme achieves an effective diversity equivalent to FH/MFSK with Time Diversity $L=w-t+1$ repetitions per symbol, while the transmission rate improvement over FH/MFSK increases proportional to $M^{1-1/t}$. We analyze here the effect of the background channel noise on the performance of this antijamming scheme.

A Steiner t-Design, denoted by $S(t,w,v)$, is a collection of blocks each one formed by the arrangement of w elements from a set $V=\{1,2,...,v\}$. These blocks are arranged such that a group of $t<w$ elements of V occurs together in at most a single block, and thus the number of common elements between two blocks in $S(t,w,v)$ is $j<t$. It can be proved that the total number of blocks in $S(t,w,v)$ is given by,

$$b = 1 + \sum_{j=0}^{t-1} x_j = \binom{v}{t} \Big/ \binom{w}{t} = \frac{v(v-1)...(v-t+1)}{w(w-1)...(w-t+1)} \tag{1}$$

where x_j is the number of blocks in $S(t,w,v)$ with exactly j common elements to a given block. It is easy to prove that the size of the set V required to represent a M-ary alphabet increases proportional to $M^{1/t}$, in contrast with M elements required by a conventional FH/MFSK scheme to represent the same alphabet. Therefore, the non-orthogonality between blocks in the Steiner set is used here to provide an effective diversity, without degrading the transmission rate.

Fig. 1 shows a simplified block diagram of a Steiner FH/MFSK System. Several non-overlapping frequency bands are available to the transmitter, each one divided into v orthogonal sub-bands. An encoder maps the input bits into a frame derived from a block in $S(t,w,v)$. The modulator consists of w synthesizers whose central frequencies are determined by independent pseudorandom sequences. Assuming that a partial band Gaussian noise jammer with variance J_0 interferes one of the w slots with probability ρ, the symbol error probability is,

$$P_e(\rho\gamma_J,\gamma_N,t,w,v) = \sum_{i=0}^{w} \binom{w}{i} \rho^i(1-\rho)^{w-i}\, P_e(\rho\gamma_J,\gamma_N,t,w,v\,|\,i) \tag{2}$$

where $P_e(\rho\gamma_J,\gamma_N,t,w,v\,|\,i)$ is the symbol error probability when exactly i slots are interfered. γ_J and γ_N are the signal-to-jammer ratio, and the signal-to-noise ratio respectively. The conditional symbol error probability can be upper bounded by,

$$P_e(\rho\gamma_J,\gamma_N,t,w,v\,|\,i) \leq \sum_{j=0}^{t-1} \sum_{k=0}^{j} x_j^k\, P_2(\gamma_{w-j,i-k}, w-j) \tag{3}$$

where x_j^k denotes the number of blocks in $S(t,w,v)$ that intersect in exactly j elements with a given block B, and also intersect with $R\subseteq B$ in exactly k elements. P_2 is the pairwise decision error probability between two decision variables formed by a non-coherent MFSK receiver [4], and $\gamma_{w-j,i-k}$ is defined as,

$$\gamma_{w-j,i-k} = (w-j-i+k)\gamma_N + (i-k)\,\frac{\rho\gamma_J\gamma_N}{\gamma_N+\rho\gamma_J} \tag{4}$$

Finally, the worst case symbol error probability is obtained maximizing the symbol error probability over ρ. Results are shown in Table 1, where the performance of a Steiner FH/MFSK based on $S(t,w,v)$, for t=2, is compared to a conventional FH/BFSK with optimal time diversity L and using nonlinear (square law) combining [5]. It is observed that for high γ_N the system performance is almost equivalent to a jammer acting on a noiseless channel, and therefore an effective diversity is gained by using multiple tones. In general, significant improvements are obtained over an uncoded FH scheme, while marginal improvements are observed over a coded FH scheme using a Hamming (7,4) code. Future research will be focused on analyzing the performance of this antijamming scheme using non-linear combining techniques (clipping, AGC) at the receiver.

Scheme	γ_N= 15 dB	γ_N=30 dB	$\gamma_N \rightarrow \infty$
Uncoded FH/BFSK	24.70 (L=3)	15.79 (L=9)	15.66 (L=10)
Hamming (7,4)	22.28 (L=2)	15.40 (L=5)	15.31 (L=5)
Steiner S(2,3,15)	28.00	27.20	26.60
Steiner S(2,4,25)	21.80	20.40	19.30
Steiner S(2,5,41)	18.20	15.10	14.80

Table 1: Signal-to-Jammer ratio required to obtain a bit error probability 10^{-5}. (The optimal diversity is indicated in parenthesis)

REFERENCES

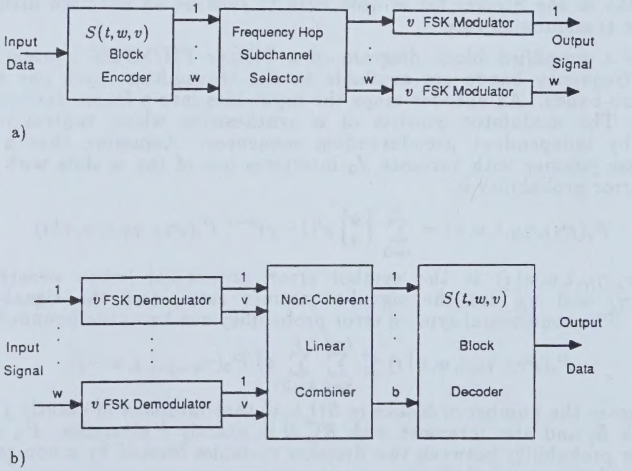
[1] Simon, M.K., Omura, J.K., Scholtz, R.A., and Levitt, B.K., *Spread Spectrum Communications*, Rockville MD, Computer Science Press, 1985.

[2] Hall, M.Jr., *Combinatorial Theory*, Blaisdell, Waltham, Mass, 1967.

[3] Atkin, G.E., and Corrales, H.P., "Performance Analysis of a Non-Orthogonal FH/MFSK System Under Partial Band Interference," *Advances on Communications, and Control Conference*, Baton Rouge, October 1988.

[4] Proakis, J.G., *Digital Communications*, McGraw-Hill Book Co., New York, 1983.

[5] Lee, J.S., French F.H., and Miller L.E., "Error-Correcting Codes and Nonlinear Diversity Combining Against Worst Case Partial-Band Noise Jamming of Frequency-Hopping MFSK Systems," IEEE Trans. on Communications, vol. 36, No. 4, April 1988, pp. 471-478.



Input
Data
S(t,w,v)
Block
Encoder
1
w
Frequency Hop
Subchannel
Selector
1
w
v FSK Modulator
v FSK Modulator
1
w
Output
Signal
a)
Input
Signal
1
w
v FSK Demodulator
v FSK Demodulator
1
v
1
v
Non-Coherent
Linear
Combiner
1
b
S(t,w,v)
Block
Decoder
Output
Data
b)

Figure 1. Steiner FH/MFSK System. a) Transmitter, b) Receiver.

PROBABILITY OF ERROR BOUNDS FOR MINIMUM SHIFT KEYING AND GAUSSIAN MINIMUM SHIFT KEYING MODULATION IN FREQUENCY SELECTIVE CHANNELS

A. S. Amanullah

CDSP Lab, Northeastern University, 360 Huntington Avenue, Boston MA 02115.

An universal portable communication system has been proposed by Cox[1] that can provide both voice and data communication in an efficient, secure, reliable, and economical manner. A suitable digital modulation scheme for portable radio applications is Gaussian Minimum Shift Keying (GMSK) which was first proposed by Murata and Hirade[2] and which is derived from MSK.

Here, we find bounds on the probability of error in frequency selective fading channels such as the one in portable radio where GMSK and MSK modulations are used. Error bounds for DPSK modulation have been found by Garber and Pursely[3]. This bound is termed the one-shot error probability and is a very good approximation to the average probability of error for small signal-to-noise ratio. Different models of delay spectra that characterize the frequency selective channels are considered.

Angle Variation: For conventional MSK, the waveform phase angle $\theta(t)$ varies linearly with time within a bit period, $\theta(t) = 2\pi f_d t$. f_d is the frequency deviation between the '0' and '1' bit.

The angle for GMSK follows a complex pattern because of the premodulation filter introduced in front of the modulator of MSK.

$$\begin{aligned}\theta(t) = {} & \pi f_d \sigma\sqrt{2}\Big(\Big[\frac{t}{\sigma\sqrt{2}} erf\left(\frac{t}{\sigma\sqrt{2}}\right) + \frac{1}{\sqrt{\pi}} e^{-\frac{t^2}{2\sigma^2}} - \frac{1}{\sqrt{\pi}}\Big] \\ & - \pi f_d\Big[\frac{t-T}{\sigma\sqrt{2}} erf\left(\frac{t-T}{\sigma\sqrt{2}}\right) + \frac{1}{\sqrt{\pi}} e^{-\frac{(t-T)^2}{2\sigma^2}} - \frac{-T}{\sigma\sqrt{2}} erf\left(\frac{-T}{\sigma\sqrt{2}}\right) - \frac{1}{\sqrt{\pi}} e^{-\frac{T^2}{2\sigma^2}}\Big]\Big)\end{aligned} \tag{1}$$

where σ is the variance of the Gaussian pre-modulation filter and T is the bit period.

Second moments: At the reciever, coherent detection is used. In [4], Bello and Nelin show that the error probability is dependent upon the second moment of the output of the matched filter at the receiver. For the evaluation of one-shot error probabilty, only a single pulse need be considered and hence ISI terms are absent in our expressions. If $\phi_c(\tau)$ represents the delay spectra for the fading channel, with τ being the delay variable and t,χ the time variable, then the moments for MSK and GMSK are respectively given by

$$M_{XX^*} = \frac{4T^2}{\pi^2} \int_0^T \phi_c(\tau)\left[1 + cos\left(\frac{\pi\tau}{T}\right)\right] d\tau. \tag{2}$$

and

$$\begin{aligned}Re\{M_{XX^*}\} = {} & \int_{-T}^{T}\int_{\tau}^{T}\int_{\tau}^{T} \phi_c(\tau) cos\Big(\pi f_d\Big[\frac{t-\tau}{\sqrt{2}} erf\left(\frac{t-\tau}{\sqrt{2}}\right) + \frac{1}{\sqrt{\pi}} e^{-\frac{(t-\tau)^2}{2}} \\ & - \frac{t-\tau-2}{\sqrt{2}} erf\left(\frac{t-\tau-2}{\sqrt{2}}\right) - \frac{1}{\sqrt{\pi}} e^{-\frac{(t-\tau-2)^2}{2}} - \frac{\chi-\tau}{\sqrt{2}} erf\left(\frac{\chi-\tau}{\sqrt{2}}\right) - \frac{1}{\sqrt{\pi}} e^{-\frac{(\chi-\tau)^2}{2}} \\ & + \frac{\chi-\tau-2}{\sqrt{2}} erf\left(\frac{\chi-\tau-2}{\sqrt{2}}\right) + \frac{1}{\sqrt{\pi}} e^{-\frac{(\chi-\tau-2)^2}{2}} + \frac{t}{\sqrt{2}} erf\left(\frac{t}{\sqrt{2}}\right) + \frac{1}{\sqrt{\pi}} e^{-\frac{(t)^2}{2}} \\ & - \frac{t-2}{\sqrt{2}} erf\left(\frac{t-2}{\sqrt{2}}\right) - \frac{1}{\sqrt{\pi}} e^{-\frac{(t-2)^2}{2}} - \frac{\chi}{\sqrt{2}} erf\left(\frac{\chi}{\sqrt{2}}\right) - \frac{1}{\sqrt{\pi}} e^{-\frac{(\chi)^2}{2}} \\ & + \frac{\chi-2}{\sqrt{2}} erf\left(\frac{\chi-2}{\sqrt{2}}\right) + \frac{1}{\sqrt{\pi}} e^{-\frac{(\chi-2)^2}{2}}\Big]\Big) dt\, d\chi\, d\tau\end{aligned} \tag{3}$$

The one-shot probability of error P_{os} for a frequency selective channel is then given by $P_{os} = \frac{1}{2}\left[1 - \sqrt{\frac{SNR_{mod}}{1+SNR_{mod}}}\right]$ where $SNR_{mod} = SNR * M_{XX^*}/T^2$. Figures 1 thru 4 are the plot for the one-shot error probability for MSK, DPSK and GMSK for the four delay spectra models under our consideration. In all the graphs, we see that the one-shot error probability for GMSK is the lowest followed by that for MSK and DPSK.

Acknowledgment The author would like to thank Dr. C.M. Keller for advice.

REFERENCES

[1] D.C. Cox, H.W. Arnold, and P.T. Porter, "Universal Digital Portable Communications: A system Perspective," *IEEE Journal on Selected Areas on Communication*, vol.SAC-5, no.5, pp.764-772, June 1987.

[2] K. Murota and K. Hirade, "GMSK Modulation for Digital Mobile Radio Telephony," *IEEE Transactions on Communications*, vol. COM-29, pp.1044-1050, July 1981.

[3] F.D. Garber and M.B. Puresly, "Performance of Differentially Coherent Digital Communications Over Frequency Selective Fading Channels, " *IEEE Transactions on Communications*, vol. 36, pp. 21-31 Jan. 1988.

[4] P.A. Bello and B.D. Nelin, "The Effect of Frequency Selective Fading on the Binary Error Probabilities of Incoherent and Differentially Coherent Matched Filter Receivers," *IEEE Transaction on Communication Systems*, vol. CS-11, pp.170-186, June 1963.

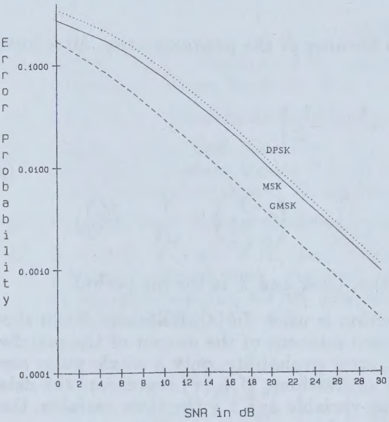


Fig. 1 One-shot Probability of Error for rectangular pulse with DPSK, GMSK and MSK modulations and Gaussian delay spectrum

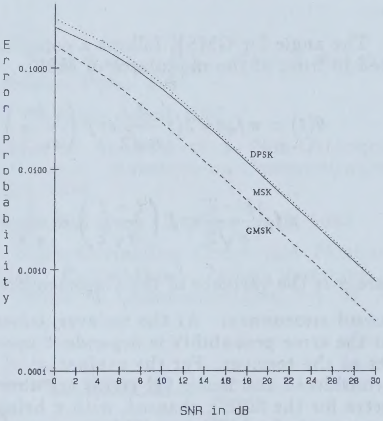


Fig. 2 One-shot Probability of Error for rectangular pulse with DPSK, GMSK and MSK modulations and exponential delay spectrum

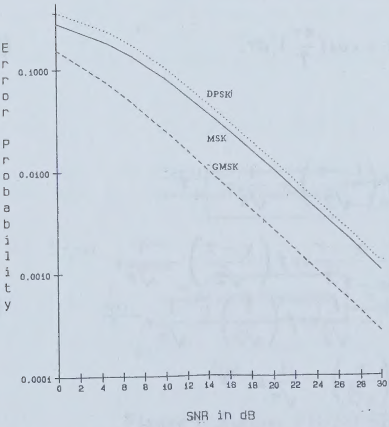


Fig. 3 One-shot Probability of Error ior rectangular pulse with DPSK, GMSK and MSK modulations and rectangular delay spectrum

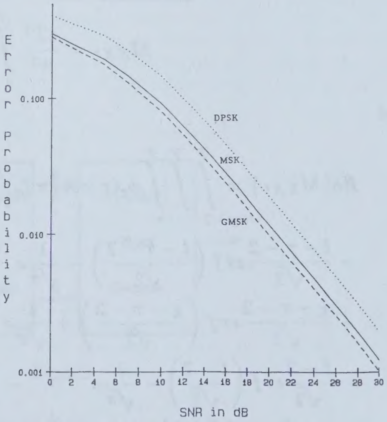


Fig. 4 One-shot Probability of Error for rectangular pulse with DPSK, GMSK and MSK modulations and equal amplitude 2-ray delay spectrum

MULTIPLE ACCESS SCHEMES WITH CONTENTION

SEUNG M. LEE* and DOUGLAS W. BROWNING
School of Electrical Engineering
Georgia Institute of Technology
Atlanta GA 30332

Abstract

An application of coding techniques to channel sensing in a multiple access environment is analyzed. We combine a code division multiple access(CDMA) technique and a carrier sensing strategy to make efficient use of the channel. Code sequences are used only for channel access and not for packet transmission. When an idle period begins, one of a small number of code sequences is randomly assigned to each *active user*. If there is at least one user with a code sequence not doubly assigned, a successful transmission follows. Throughput is analyzed for slotted access systems and the results show improvement over conventional channel sensing techniques.

1 INTRODUCTION

Since its developement in the 1940's, most applications of spread spectrum techniques [1] have been confined to military problems such as Antijam(AJ) protection or Low Probability of Intercept(LPI) techniques. Recently, however, CDMA techniques have been applied to multiple access communication systems [2,3,4,5]. While the overlap of packets cause collisions in conventional multiple access schemes, simultaneous transmission of packets is possible in CDMA systems with a proper choice of code sequences.

In applying CDMA, there are many possible channel access protocols. A basic issue is whether to use a single code sequence or a set of different code sequences. When a single code sequence is used, a sufficient time offset is required between different transmissions to distinguish them from each other. Lee and Un [6] used a time-shifted version of a single maximal length sequence for a CDMA LAN where the address of a transceiver is represented by the amount of time shifting. But as the propagation delay becomes small, the users do not maintain a sufficient offset and the performance of the system becomes greatly degraded.

When a set of different code sequences are used, sequences with low cross correlations such as Gold sequences [7] should be used.

In a receiver-oriented protocol, a set of codes is assigned to each receiver and the code itself serves as a destination address. This protocol was analyzed by Davis and Gronemeyer [8] and the implementation was discussed by Kahn [9]. The disadvantage of this protocol is that when two or more users are sending with the same sequence and there is insufficient time offset, the receiver cannot distinguish among them and a collision occurs.

In a transmitter-oriented protocol, each transmitter has its own unique code sequence. Raychaudhuri [5] evaluated the performance of a synchronous transmitter-oriented packet radio network. The drawback of this scheme is that the receiver has to have information on all the code sequences used in the system and this can increase the complexity of the receiver.

The above protocols can be combined together according to the environment. Silvester and Sousa [10] proposed common-transmitter hybrid and receiver-transmitter hybrid protocols which involve changing the code sequence of a transmission after an initial header is transmitted.

We combine CDMA techniques and carrier sensing techniques for channel acquisition. The following terms are used.

- An *active user* is a user who has a packet ready to transmit.

---

*This research was supported by a grant from AT&T

- A *survivor* is a user whose code sequence is assigned to it only and not to any other users during a *contention period.*
- A system is in a *contention period* if all the *active users* are sending their code sequences to determine the *survivor.*
- A system is in a *busy period* if any user is transmitting a packet successfully or if the system is in a *contention period.*
- A system is in an *idle period* if there are no transfers of packets between users.
- A system suffers a *collision* if one code sequence is assigned to at least two users, and no other code sequence is assigned to exactly one user.

In our system the active users are each randomly assigned one of a limited set of predetermined code sequences and are required to wait until the channel becomes idle. Due to the limited number of code sequences n, the same code sequence may be assigned to several active users. During a contention period each active user sends its code sequence and monitors the network to see which codes are being used by other active users. This can be done by loading a local correlator or matched filter with a copy of the codes. Assume that if any user code sequence is sent by only one user, that user becomes a survivor. If there is at least one survivor, a successful transmission follows.

We consider three schemes for handling multiple survivors. First, we can send *only one* survivor packet regardless of the number of survivors in an order determined from the priority of each code sequence (single-success system). Second we can send *all* survivor packets in an order determined from the priority of each code sequence (multi-success system). Finally a variation of the latter method is to send *up to only a pre-determined number* of the survivor packets (L-success system).

2 ANALYSIS

2.1 Single Success System

In this system, only a single packet is sent even if there are multiple survivors. The priority is determined according to a pre-specified order of the codes.

2.1.1 System Descriptions

Define A_n^k as the number of code assignments where at least one survivor exists with n code sequences and k active users in the system. If each user chooses a code randomly, the probability that at least one survivor exists,a_n^k, can be expressed as

$$a_n^k = A_n^k / n^k \tag{1}$$

A_n^k can be obtained from the following recursive formula.

$$A_n^k = n\{(n-1)^{k-1} + \sum_{i=1}^{k-2} \binom{k-1}{i} A_{n-1}^{k-i-1}\} \tag{2}$$

This can be derived as follows. Consider one of the code sequences. If that code sequence is used by only one user then that user becomes a survivor and the system is successful. Therefore the other $(n-1)$ sequences can be assigned arbitrarily to the remaining $(k-1)$ users, so the number of cases for those events is $(n-1)^{k-1}$. Next, if that code sequence is occupied by $(i+1)$ users $(i = 1, 2, \cdots, k-2)$ then that sequence and $(i+1)$ users are excluded and our system has $(n-1)$ sequences and $(k-i-1)$ users remaining. Then the number of cases of success for each i can be written as $\binom{k-1}{i} A_{n-1}^{k-i-1}$. Since this holds for each sequence, we obtain Equation 2 by multiplying by n.

The probability a_n^k is then given as the following.

$$a_n^k = (1 - \frac{1}{n})^{k-1} + \frac{1}{n^{k-1}} \sum_{i=1}^{k-2} \binom{k-1}{i} A_{n-1}^{k-1-i} \tag{3}$$

To analyze the effect of our scheme on conventional random access methods, we first have to find the probability of successful transmission P_s,. This is the probability of at least one survivor during one contention period, given that some active users are present.

$$
\begin{aligned}
P_s &= \Pr[\text{at least one survivor in slot} \mid \text{some arrivals}] \\
&= \frac{\Pr[\text{at least one survivor in slot and some arrivals}]}{\Pr[\text{some arrivals}]} \\
&= \frac{\sum_{k=1}^{\infty} a_n^k P_k}{1 - P_0} \qquad (4)
\end{aligned}
$$

where P_k is the probability of k arrivals. The number of arrivals has a Poisson distribution with mean aG, where G is the offered traffic and a is the normalized end-to-end propagation delay.

2.1.2 Throughput Analysis

To find the throughput S, we use a well-known result from renewal processes[11]

$$S = \frac{\bar{U}}{\bar{I} + \bar{B}} \qquad (5)$$

where I is the length of an idle period, B is the length of a busy period, U is the time spent in successful transmission and $\bar{X}$ represents the expected value of the random variable X. We consider slotted non-persistent and 1-persistent access. For the non-persistent case, if the packet arrives during an idle period it sends the code sequence at the beginning of the next slot, and if it arrives during a busy period the transmission is rescheduled. In the 1-persistent case, if the packet arrives during the busy period, it waits until the next idle period or contention period begins.

For the non-persistent case, $\bar{U}$ is given by

$$\bar{U} = P \cdot P_s \qquad (6)$$

where P is the packet length and P_s is obtained from Equation 4. The busy period $\bar{B}$ is given by

$$
\begin{aligned}
\bar{B} &= P_s(P + \acute{\gamma} + \tau) + (1 - P_s)(\acute{\gamma} + \tau) \\
&= P_s \cdot P + \acute{\gamma} + \tau \qquad (7)
\end{aligned}
$$

where $\acute{\gamma}$ is the time spent checking the code sequence.

The idle period I is geometrically distributed for a slotted system and its mean is given by

$$\bar{I} = \frac{\tau e^{-aG}}{1 - e^{-aG}} \qquad (8)$$

Using the above equations we can compute the throughput of our scheme as

$$S = \frac{P_s}{(P_s + a + \gamma)(1 - e^{-aG}) + a e^{-aG}} \qquad (9)$$

where γ is the normalized value of $\acute{\gamma}$, $\gamma = \frac{\acute{\gamma}}{P}$.

For the 1-persistent case, throughput can also be obtained using Equation 5. To find $\bar{B}$ and $\bar{U}$, we follow the analysis used by Tobagi and Hunt [12]. In the 1-persistent CSMA/CD scheme the success or failure of a transmission period depends on the preceding transmission period except for the first busy period. If the previous period is of length X slots, the average duration of the busy period or average successful transmission time following a busy period are given as a function of X and are denoted as $B(X)$ and $U(X)$. Then the average busy period $\bar{B}$ is given by $B(1)$ and $\bar{U}$ is given by $U(1)$. Using Equation 5, throughput can be obtained as the following.

$$S = \frac{U(1)}{B(1) + \frac{a}{1 - e^{-aG}}} \qquad (10)$$

In our scheme $X = 1 + \frac{1}{a} + \frac{\gamma}{a}$ for a successful transmission and $X = 1 + \frac{\gamma}{a}$ for an unsuccessful transmission. Then from[12], we have

$$
\begin{aligned}
B(X) &= P_s(X)(1+a+\gamma+(1-P_0(1+\frac{1}{a}+\frac{\gamma}{a}))B(1+\frac{1}{a}+\frac{\gamma}{a})) \\
&\quad +(1-P_s(X))(a+\gamma+(1-P_0(1+\frac{\gamma}{a}))B(1+\frac{\gamma}{a})) \qquad (11) \\
U(X) &= P_s(X)(1+(1-P_0(1+\frac{1}{a}+\frac{\gamma}{a}))U(1+\frac{1}{a}+\frac{\gamma}{a})) \\
&\quad +(1-P_s(X))(1-P_0(1+\frac{\gamma}{a}))B(1+\frac{\gamma}{a}) \qquad (12)
\end{aligned}
$$

Using $X = 1+\frac{1}{a}+\frac{\gamma}{a}$ and $X = 1+\frac{\gamma}{a}$ we can solve above equation for $B(1+\frac{1}{a}+\frac{\gamma}{a})$ and $B(1+\frac{\gamma}{a})$. From these we can then obtain $B(1)$. The same procedure can be followed in finding $U(1)$.

2.1.3 Delay Analysis

Another important parameter for performance is the expected delay D, which is defined as the average time from the generation of a packet until it is successfully received. We can roughly estimate the delay characteristics using the term G/S , which is the average number of transmissions required per successful transmission. To find more accurate results we use a linear feedback model with an imbedded Markov chain used in analyzing CSMA and CSMA/CD schemes by Tobagi [13] and by Hunt [12].

In the imbedded Markov chain model, there are M users and every user belongs to either an idle state or a *backlogged* state. In the idle state, users can generate a new packets with probability σ. If they collide or find the channel busy they switch to the *backlogged* state. In the *backlogged* state, they retransmit the packet with probability ν in each slot and switch to the idle state when a retransmission is successful.

Let N^t be a random variable representing the number of backlogged users at the beginning of slot t. We call the first slot of each idle period an imbedded slot and denote its time of occurrence as t_e. I is the length of idle period, and T is the packet transmission time in slots. We proceed to find a stationary probability distribution

$$\pi = (\pi_0, \pi_1, \cdots, \pi_M) \qquad (13)$$

where $\pi_j = \lim_{t_e \to \infty}$ Pr[$N^{t_e} = j$]. The vector π can be obtained using the equation

$$\pi = \pi P \qquad (14)$$

where P is the transition matrix of the imbedded Markov chain. P can be derived as follows.

Let R be the transition matrix between slot (t_e+I-1) and (t_e+I). Its elements can be described as

$$r_{ik} = \Pr[\ N^{t_e+I} = k \mid N^{t_e+I-1} = i] \qquad (15)$$

The matrix R can be written as sum of the matrices C and F where C and F have elements

$$
\begin{aligned}
c_{ik} &= \Pr[\ N^{t_e+I} = k \text{ and transmission is successful} \mid N^{t_e+I-1} = i] \\
&= \Pr[\ n = k-i \text{ and at least 1 survivors} \mid N^{t_e+I-1} = i] \\
f_{ik} &= \Pr[\ N^{t_e+I} = k \text{ and transmission is unsuccessful} \mid N^{t_e+I-1} = i] \\
&= \Pr[\ n = k-i \text{ and no survivors} \mid N^{t_e+I-1} = i] \qquad (16)
\end{aligned}
$$

where n represents the number of new packets generated by $(M-i)$ idle stations. Let Q be the transition matrix for all remaining slots. Its elements can be written as

$$q_{ik} = \Pr[\ N^t = k \mid N^{t-1} = i] \qquad (17)$$

The length of the busy period is $(T+\gamma+1)$ for successful transmissions and $(\gamma+1)$ for unsuccessful transmissions, therefore the system transition matrix P can be written as

$$P = CQ^{T+\gamma+1}J + FQ^{\gamma+1} \qquad (18)$$

where J represents the fact that a successful transmission decreases the backlog by 1, and is defined as

$$j_{ik} = \begin{cases} 1 & k = i-1 \\ 0 & \text{otherwise.} \end{cases} \qquad (19)$$

Assuming that $N^{t_e+I-1} = i$, the elements of C, F and Q can be described as the following.

$$c_{ik} = \begin{cases} 0 & k < i \\ \frac{\binom{M-i}{k-i}\sigma^{k-i}(1-\sigma)^{M-k}\{\sum_{j=0}^{i}\binom{i}{j}\nu^j(1-\nu)^{i-j}a_N^{j+k-i}\}}{1-(1-\sigma)^{M-i}(1-\nu)^i} & k \geq i \end{cases} \tag{20}$$

$$f_{ik} = \begin{cases} 0 & k < i \\ \frac{(1-\sigma)^{M-i}\{1-(1-\nu)^i-\sum_{j=1}^{i}\binom{i}{j}\nu^j(1-\nu)^{i-j}a_N^j\}}{1-(1-\sigma)^{M-i}(1-\nu)^i} & k = i \\ \frac{\binom{M-i}{k-i}\sigma^{k-i}(1-\sigma)^{M-k}\{1-\sum_{j=0}^{i}\binom{i}{j}\nu^j(1-\nu)^{i-j}a_N^{j+1}\}}{1-(1-\sigma)^{M-i}(1-\nu)^i} & k \geq i+1 \end{cases} \tag{21}$$

$$q_{ik} = \begin{cases} 0 & k < i \\ \binom{M-i}{k-i}\sigma^{k-i}(1-\sigma)^{M-k} & k \geq i \end{cases} \tag{22}$$

Let $P_s(i)$ be the probability of successful transmission during a cycle with $N^{t_e} = i$, given by

$$\begin{aligned} P_s(i) &= \sum_{k=i}^{M-i} c_{ik} \\ &= \sum_{k=i}^{M-i} \binom{M-i}{k-i}\sigma^{k-i}(1-\sigma)^{M-k}\sum_{j=0}^{i}\{\binom{i}{j}\nu^j(1-\nu)^{i-j}a_N^{j+k-i}\} \end{aligned} \tag{23}$$

Let δ_i be the probability that no user is ready during slot t given $N^t = i$. Then

$$\delta_i = (1-\sigma)^{M-i}(1-\nu)^i \tag{24}$$

The average length of an idle period is given by $\frac{1}{1-\delta_i}$.
So the average channel throughput S_{out} can be obtained as

$$S_{\text{out}} = \frac{\sum_{i=0}^{M}\pi_i P_s(i)T}{\sum_{i=0}^{M}\pi_i\{\frac{1}{1-\delta_i} + P_s(i)(1+\gamma+T) + (1-P_s(i))(1+\gamma)\}} \tag{25}$$

Using the same method for average channel backlog $\bar{N}$ we have

$$\bar{N} = \frac{\sum_{i=1}^{M}\pi_i\{\frac{i}{1-\delta_i} + A(i)\}}{\sum_{i=1}^{M}\pi_i\{\frac{1}{1-\delta_i} + P_s(i)(1+\gamma+T) + (1-P_s(i))(1+\gamma)\}} \tag{26}$$

where $A(i)$ is the expected sum of backlogs over all slots in the busy period with $N^{t_e} = i$, given by

$$\begin{aligned} A(i) &= \sum_{m=0}^{T+\gamma}\sum_{j=i}^{M} j[CQ^m]_{ij} + \sum_{m=0}^{\gamma}\sum_{j=i}^{M} j[FQ^m]_{ij} \\ &= \sum_{j=i}^{M} j[C\sum_{m=0}^{T+\gamma}Q^m + F\sum_{m=0}^{\gamma}Q^m]_{ij} \end{aligned} \tag{27}$$

By Little's law[14] the average packet delay D is given by

$$D = \frac{\bar{N}}{S_{\text{out}}} \tag{28}$$

2.2 Multi-success System

In this section, we consider the case of a multi-success system. In this case, all survivors are sent in order after each contention period.

2.2.1 System Description

We can use the same approach as for a single success system by extending the concept of the probability of successful transmission P_s.

We use the following notation with n code sequences and k active users,

$$\begin{aligned}
A_n^k(i) &= [\text{number of code assignments where at least } i \text{ survivors exist}] \\
a_n^k(i) &= \Pr[\text{at least } i \text{ survivors exist}] \\
&= A_n^k(i)/n^k \\
O_n^k(i) &= [\text{number of code assignments where only } i \text{ survivors exist}] \\
o_n^k(i) &= \Pr[\text{only } i \text{ survivors exist}] \\
&= O_n^k(i)/n^k
\end{aligned} \tag{29}$$

Using this notation, we can extend the notation in the previous section as

$$\begin{aligned}
A_n^k &= A_n^k(1) \\
a_n^k &= a_n^k(1)
\end{aligned} \tag{30}$$

Extending the method of finding $a_n^k(1)$ in the previous section we can write $A_n^k(i)$ as follows.

$$\begin{aligned}
A_n^k(1) &= n\{(n-1)^{k-1} + \sum_{j=1}^{k-2} \binom{k-1}{j} A_{n-1}^{k-1-j}(1)\} \\
A_n^k(2) &= n\{A_{n-1}^{k-1}(1) + \sum_{j=1}^{k-3} \binom{k-1}{j} A_{n-1}^{k-1-j}(2)\} \\
&\cdots \\
A_n^k(n) &= n\{A_{n-1}^{k-1}(n-1) + \sum_{j=1}^{k-j-1} \binom{k-1}{j} A_{n-1}^{k-1-j}(n)\}
\end{aligned} \tag{31}$$

We can write $O_n^k(i)$ as

$$\begin{aligned}
O_n^k(i) &= A_n^k(i) - A_n^k(i+1) & i < n \\
O_n^k(n) &= A_n^k(n)
\end{aligned} \tag{32}$$

To analyze the system we must find the probabilty of success P_s, which can be written as

$$P_s = P_{s_1} + P_{s_2} + \cdots + P_{s_n} \tag{33}$$

where P_{s_i} is

$$\begin{aligned}
P_{s_i} &= \Pr[\text{only } i \text{ survivors in slot} \mid \text{some arrivals occur}] \\
&= \frac{1}{1-P_0} \sum_{k=i}^{\infty} o_n^k(i) P_k \qquad (i = 1, 2, \cdots n)
\end{aligned} \tag{34}$$

2.2.2 Throughput Analysis

To find the throughput characteristic S for the non-persistent system, we use the same approach in the previous section. $\bar{U}$ is given by

$$\bar{U} = P \cdot P_{s_1} + 2P \cdot P_{s_2} + \cdots + (n-1)P \cdot P_{s_{n-1}} + nP \cdot P_{s_n} = (\sum_{i=1}^{n} i P_{s_i})P \tag{35}$$

The busy period $\bar{B}$ is given by

$$\begin{aligned}
\bar{B} &= P_{s_1}(P + \tau + \acute{\gamma}) + P_{s_2}(2P + 2\tau + \acute{\gamma}) + \cdots + P_{s_n}(nP + n\tau + \acute{\gamma}) + (1 - P_s)(\tau + \acute{\gamma}) \\
&= \sum_{i=1}^{n} P_{s_i}(iP + i\tau + \acute{\gamma}) + (1 - P_s)(\tau + \acute{\gamma})
\end{aligned} \tag{36}$$

Idle period $\bar{I}$ is given by

$$\bar{I} = \frac{\tau e^{-aG}}{1 - e^{-aG}} \tag{37}$$

Using the above equations we can compute the throughput of a non-persistent system as

$$S = \frac{\sum_{i=1}^{n} i P_{s_i}}{\sum_{i=1}^{n} P_{s_i}(i + ia + \gamma) + (1 - P_s)(a + \gamma) + \frac{ae^{-aG}}{1-e^{-aG}}} \tag{38}$$

For the 1-persistent system, we have

$$\begin{aligned} B(X) = \quad & P_{s_1}(X)(1 + a + \gamma + (1 - P_0(1 + \tfrac{1}{a} + \tfrac{\gamma}{a}))B(1 + \tfrac{1}{a} + \tfrac{\gamma}{a})) \\ & + P_{s_2}(X)(2 + 2a + \gamma + (1 - P_0(2 + \tfrac{2}{a} + \tfrac{\gamma}{a}))B(2 + \tfrac{2}{a} + \tfrac{\gamma}{a})) \\ & \cdots \\ & + P_{s_n}(X)(n + na + \gamma + (1 - P_0(n + \tfrac{n}{a} + \tfrac{\gamma}{a}))B(n + \tfrac{n}{a} + \tfrac{\gamma}{a})) \\ & + (1 - P_s(X))(a + \gamma + (1 - P_0(1 + \tfrac{\gamma}{a}))B(1 + \tfrac{\gamma}{a})) \end{aligned} \tag{39}$$

$$\begin{aligned} U(X) = \quad & P_{s_1}(X)(1 + (1 - P_0(1 + \tfrac{1}{a} + \tfrac{\gamma}{a}))U(1 + \tfrac{1}{a} + \tfrac{\gamma}{a})) \\ & + P_{s_2}(X)(2 + (1 - P_0(2 + \tfrac{2}{a} + \tfrac{\gamma}{a}))U(2 + \tfrac{2}{a} + \tfrac{\gamma}{a})) \\ & \cdots \\ & + P_{s_n}(X)(n + (1 - P_0(n + \tfrac{n}{a} + \tfrac{\gamma}{a}))U(n + \tfrac{n}{a} + \tfrac{\gamma}{a})) \\ & + (1 - P_s(X))(1 - P_0(1 + \tfrac{\gamma}{a}))U(1 + \tfrac{\gamma}{a}) \end{aligned} \tag{40}$$

$$\bar{I} = \frac{a}{1 - e^{-aG}} \tag{41}$$

We substitute $X = 1 + \frac{\gamma}{a}, 1 + \frac{1}{a} + \frac{\gamma}{a}, \cdots, n + \frac{n}{a} + \frac{\gamma}{a}$ for X in Equation 39 for B. After solving for $B(1 + \frac{\gamma}{a}), B(1 + \frac{1}{a}), \cdots, B(n + \frac{n}{a} + \frac{\gamma}{a})$ and we can then obtain $B(1)$. The same procedure can be followed to find $U(1)$.

Then the throughput is given by

$$S = \frac{U(1)}{B(1) + \frac{a}{1-e^{-aG}}} \tag{42}$$

Using Equation 42, we can obtain the relation between channel capacity $S_{\max}$ ($S_{\max}$ is defined as the maximum value of S over all offered loads) and propagation delay.

2.3 *L*-success System

In this system, up to L survivor packets are sent and the rest of the survivors are rescheduled.

2.3.1 System Description

The analysis of this case is the same as that for the multi-success system except for the definition of P_s, which can be written as

$$P_s = P_{s_1} + P_{s_2} + \cdots + P_{s_L} \tag{43}$$

where P_{s_i} is

$$\begin{aligned} P_{s_i} &= \text{Pr[only } i \text{ survivors are in slot } | \text{ some arrival occurs]} \\ &= \frac{1}{1 - P_0} \sum_{k=i}^{\infty} o_n^k(i) P_k \qquad (i = 1, 2, \cdots, L-1) \\ P_{s_L} &= \text{Pr[at least } L \text{ survivors are in slot and some arrival occurs]} \\ &= \frac{1}{1 - P_0} \sum_{k=L}^{\infty} a_n^k(L) P_k \end{aligned} \tag{44}$$

2.3.2 Throughput Analysis

To find the throughput for the non-persistent system, we can find U, B and I in the same manner as for the multi-success system.

$$\bar{U} = P \cdot P_{s_1} + P \cdot P_{s_2} + \cdots + (L-1)P \cdot P_{s_{L-1}} + LP \cdot P_{s_L} = (\sum_{i=1}^{L} iP_{s_i})P \tag{45}$$

$$\begin{aligned} \bar{B} &= P_{s_1}(P + \tau + \acute{\gamma}) + P_{s_2}(2P + 2\tau + \acute{\gamma}) + \cdots + P_{s_L}(LP + L\tau + \acute{\gamma}) + (1 - P_s)(\tau + \acute{\gamma}) \\ &= \sum_{i=1}^{L} P_{s_i}(iP + i\tau + \acute{\gamma}) + (1 - P_s)(\tau + \acute{\gamma}) \end{aligned} \tag{46}$$

$$\bar{I} = \frac{\tau e^{-aG}}{1 - e^{-aG}} \tag{47}$$

Then throughput is given by Equation 5.
For the 1-persistent case, we have

$$\begin{aligned} B(X) = \; & P_{s_1}(X)(1 + a + \gamma + (1 - P_0(1 + \tfrac{1}{a} + \tfrac{\gamma}{a}))B(1 + \tfrac{1}{a} + \tfrac{\gamma}{a})) \\ & + P_{s_2}(X)(2 + 2a + \gamma + (1 - P_0(2 + \tfrac{2}{a} + \tfrac{\gamma}{a}))B(2 + \tfrac{2}{a} + \tfrac{\gamma}{a})) \\ & \cdots \\ & + P_{s_L}(X)(L + La + \gamma + (1 - P_0(L + \tfrac{L}{a} + \tfrac{\gamma}{a}))B(L + \tfrac{L}{a} + \tfrac{\gamma}{a})) \\ & + (1 - P_s(X))(a + \gamma + (1 - P_0(1 + \tfrac{\gamma}{a}))B(1 + \tfrac{\gamma}{a}) \end{aligned} \tag{48}$$

$$\begin{aligned} U(X) = \; & P_{s_1}(X)(1 + (1 - P_0(1 + \tfrac{1}{a} + \tfrac{\gamma}{a}))U(1 + \tfrac{1}{a} + \tfrac{\gamma}{a})) \\ & + P_{s_2}(X)(2 + (1 - P_0(2 + \tfrac{2}{a} + \tfrac{\gamma}{a}))U(2 + \tfrac{2}{a} + \tfrac{\gamma}{a})) \\ & \cdots \\ & + P_{s_L}(X)(L + (1 - P_0(L + \tfrac{L}{a} + \tfrac{\gamma}{a}))U(L + \tfrac{L}{a} + \tfrac{\gamma}{a})) \\ & + (1 - P_s(X))(1 - P_0(1 + \tfrac{\gamma}{a}))U(1 + \tfrac{\gamma}{a}) \end{aligned} \tag{49}$$

$$\bar{I} = \frac{a}{1 - e^{-aG}} \tag{50}$$

We substitute $X = 1 + \frac{\gamma}{a}, 1 + \frac{1}{a} + \frac{\gamma}{a}, \cdots, L + \frac{L}{a} + \frac{\gamma}{a}$ for X in the above equation for B and U and obtain $B(1)$ and $U(1)$.

The throughput is given by Equation 42.

3 NUMERICAL RESULTS

In this section, we present the numerical results and compare these schemes with conventional ones.

A comparison of throughput performance with other random access schemes are shown in Figure 1 for the 1-persistent case. As we can see in Figure 1, our schemes show a higher value of throughput S than other schemes.

Using the S-G characteristic, we can obtain the relation between propagation delay a and channel capacity $S_{\max}$ which is defined as the maximum value of throughput S over all offered loads. Curves for the 1-persistent case are shown in Figure 2.

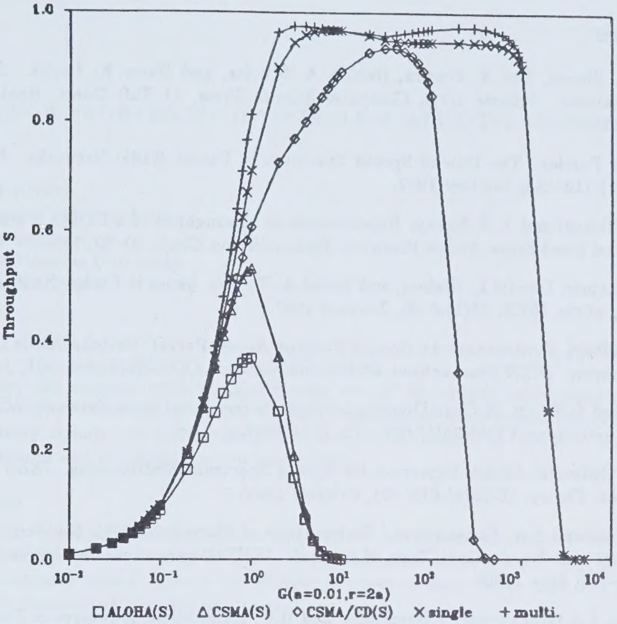


Figure 1: S versus G characteristics of ALOHA, 1-persistent CSMA, CSMA/CD, single success and multi-success schemes ($n = 7$)

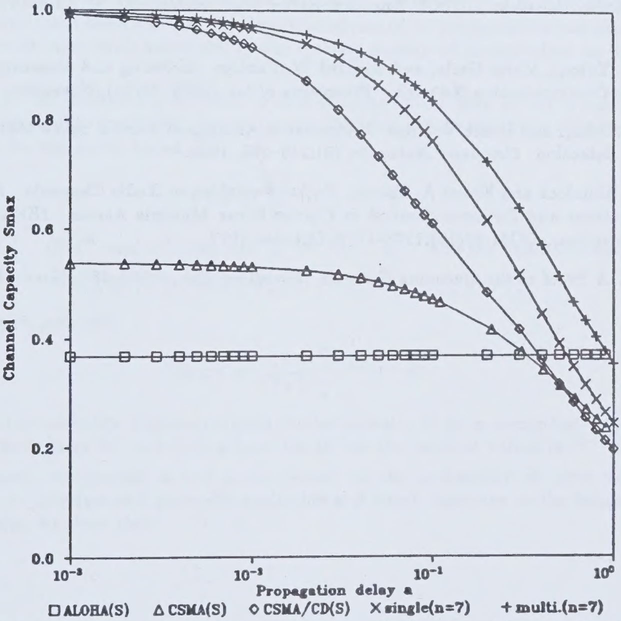


Figure 2: Comparison of maximum throughput versus normalized propagation delay for ALOHA, 1-persistent CSMA, CSMA/CD, single success and multi-success schemes ($a = 0.01, \acute{\gamma} = 2a$)

References

[1] Marvin K. Simon, Jim K. Omura, Robert A. Scholtz, and Barry K. Levitt. *Spread Spectrum Communications.* Volume 1,2,3, Computer Science Press, 11 Taft Court, Rockville, Maryland 20850, 1985.

[2] Michael B. Pursley. The Role of Spread Spectrum in Packet Radio Networks. *Proceeding of the IEEE,* 75(1):116–134, January 1987.

[3] Fouad A. Tobagi and J. S. Storey. Improvement in Throughput of a CDMA Packet Network Due to a Channel Load Sense Access Protocol. *Proc. Allerton Conf.*, 40–49, 1984.

[4] Barry M. Leiner, Donald L. Nielson, and Fouad A. Tobagi. Issues in Packet Radio Network Design. *Proceeding of the IEEE,* 75(1):6–20, January 1987.

[5] D. Raychadhuri. Performance Analysis of Random Access Packet- Switched Code Division Multiple Access Systems. *IEEE Transactions on Communications*, COM-29(6):895–901, June 1981.

[6] J.R. Lee and C.K. Un. A Code Division Multiple Access Local Area Network. *IEEE Transactions on Communications*, COM-35(6):667–671, June 1987.

[7] R. Gold. Optimum Binary Sequences for Spread Spectrum Multiplexing. *IEEE Transactions on Information Theory,* IT-13(5):619–621, October 1967.

[8] D.H. Davies and S.A. Gronemeyer. Performance of Slotted ALOHA Random Access with Delay Capture and Randomized Time of Arrival. *IEEE Transactions on Communications*, COM-80(5):703–710, May 1980.

[9] R.E.Kahn, S.A.Gronemeyer, J.Burchfield, and R.C. Kunzelman. Advances in Packet Radio Technology. *Proceeding of the IEEE,* 66(11):1468–1496, November 1978.

[10] Elvino S. Sousa and John A. Silvester. Spreading Code Protocols for Distributed Spread Spectrum Packet Radio Networks. *IEEE Transactions on Communications*, COM-36(3):272–281, March 1988.

[11] Fouad A. Tobagi, Mario Gerla, and Richard W. Pebbles. Modeling and Measurement Techniques in Packet Communication Networks. *Proceeding of the IEEE,* 66(11), November 1978.

[12] Fouad A.Tobagi and Bruce V. Hunt. Performance Analysis of Carrier Sence Multiple Access with Collision detection. *Computer Networks,* (4):245–259, 1980.

[13] Leonard Kleinlock and Fouad A. Tobagi. Packet Switching in Radio Channels : Part IV- Stability Considerations and Dynamic Control in Carrier Sense Multiple Access. *IEEE Transactions on Communications*, COM-25(10):1103–1119, October 1977.

[14] J. Little. A Proof of the Queueing Formula. *Operation Res.*, 9:383–387, March-April 1961.

ON THE ERROR PROBABILITY OF SIGNALS IN ADDITIVE GAUSSIAN NOISE

BRIAN HUGHES
Department of Electrical and
Computer Engineering
The Johns Hopkins University
Baltimore, MD 21218

ABSTRACT

We consider the classical problem of detecting one of M equally likely signals in additive white Gaussian noise. A new upper bound to the probability of error is introduced with can be applied to arbitrary signals sets and is straightforward to calculate. This bound is always tighter that the union bound and the minimum distance bound.

1. Introduction

Consider the classical problem of detecting one of M signals in additive white Gaussian noise: An integer message m, equally likely to be any element in $\{0, \ldots, M-1\}$, is transmitted over an N-dimensional vector channel by sending a corresponding vector $\mathbf{x}_0, \ldots, \mathbf{x}_{M-1} \in \mathbf{R}^N$. When $m = i$ is transmitted, the received vector is

$$\mathbf{y} = \mathbf{x}_i + \mathbf{n} , \tag{1.1}$$

where the components of $\mathbf{n}$ are independent, identically distributed $N(0,\sigma^2)$ random variables. An estimate $\hat{m}$ is made at the receiver using the maximum likelihood rule. A basic result from elementary communication theory is that the detection of one of M equiprobable waveforms in additive white Gaussian noise with two-sided power spectral density σ^2 is equivalent to the detection problem in (1.1).

Let $P_{e,0}$ be the probability of error of this detector given that $m=0$ is sent. There are several simple, generally applicable, upper bounds for this error probability. Two of the most commonly used are the *union bound*

$$P_{e,0} \leq \sum_{i=1}^{M-1} Q(d_i/2\sigma) , \tag{1.2}$$

where $Q(x) \equiv \int_x^\infty (2\pi)^{-1/2} \exp(-x^2/2)dx$ and $d_i \equiv |\mathbf{x}_i - \mathbf{x}_0|$; and the *minimum distance bound*

$$P_{e,0} \leq Pr\{ |\mathbf{n}| \geq d_{\min}/2 \} = \overline{\Gamma}(N/2, d_{\min}^2/8\sigma^2) , \tag{1.3}$$

where $d_{\min} = \min_{i \neq 0} d_i$ and

$$\overline{\Gamma}(a,x) \equiv \frac{1}{\Gamma(a)} \int_x^\infty t^{a-1} e^{-t}\, dt \tag{1.4}$$

is a (normalized) *incomplete gamma function.* Unfortunately, (1.2) is somewhat loose for small values of $d_{\min}/2\sigma$ or large M, and (1.3) is loose for all but the smallest values of N.

In this paper, we present a new upper bound on the probability of error which is also straightforward to calculate and generally applicable and which improves on the bounds (1.2) and (1.3). In particular, we show that

$$P_{e,0} \le \sum_{i=1}^{M-1} A_N(\, d_i/\alpha_o \,,\, d_i/\sqrt{2\sigma}\,) \tag{1.5}$$

where α_o is the unique solution of

$$\sum_{i=1}^{M-1} A_N(\, d_i/\alpha_o \,,\, 0\,) = 1 \,, \tag{1.6}$$

$$A_N(y\,,x) \equiv \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)\, \overline{\gamma}((N-1)/2 \,,\, z^2/2y^2)\, dz \tag{1.7}$$

and

$$\overline{\gamma}(a\,,x) \equiv 1 - \overline{\Gamma}(a\,,x)\,. \tag{1.8}$$

We also present several series expansions for $A_N(y\,,x)$ that allow rapid calculation of (1.5) and (1.6).

The remainder of this paper is organized as follows. In Section 2, we derive the bound in (1.4) and explore its properties. In Section 3, we present series expansions for $A_N(y\,,x)$ and $A_N(y\,,0)$. Finally, in Section 4, we present an example.

2. Derivation of the Bound

For the channel described in (1.1), the maximum likelihood decoder is the minimum Euclidean distance rule: [1]

$$\hat{m}(\mathbf{y}) \equiv \arg \min_{1 \le i \le M-1} |\, \mathbf{x}_i - \mathbf{y}\, | \tag{2.1}$$

with ties resolved arbitrarily. For convenience, we resolve ties in favor of the *larger* index, and define decoding sets $D_i = \{\mathbf{y} \mid \hat{m}(\mathbf{y}) = i\}$, $0 \le i \le M-1$.

The aim of this section is to derive a simple upper bound for the error probability

$$P_{e,i} \equiv Pr\{\, \mathbf{x}_i + \mathbf{n} \in D_i^c \,\}\,, \tag{2.2}$$

where $D_i^c = \mathbf{R}^N - D_i$. Without loss of generality, we take $m = 0$ in the sequel. The derivation of this bound can be conceptually divided into three stages. First, we show that D_0^c can be decomposed into a disjoint union of truncated polyhedral cones with vertices at $\mathbf{x}_0$. Second, we show that the error probability is bounded above by replacing these cones with circular cones of the same solid angle. Finally, we develop a further upper bound by maximizing over all feasible solid angles.

A Decomposition for D_0^c : The decoding sets obtained from $\hat{m}$ in (2.1) can be formed by intersecting half-spaces, as in

$$D_0 = \bigcap_{i=1}^{M-1} H_i^c\,, \tag{2.3}$$

$$H_i \equiv \{\, \mathbf{y} \in \mathbf{R}^N \mid |\mathbf{y}-\mathbf{x}_i| \le |\mathbf{y}-\mathbf{x}_0| \,\}\,. \tag{2.4}$$

As suggested by (2.3), D_0 is a polyhedron formed by at most $M-1$ supporting hyperplanes which are the boundaries of the half-spaces H_i , $1 \le i \le M-1$.

We now present a geometric decomposition of D_0^c in terms of disjoint polyhedral cones. Let $\mathbf{x} \ne \mathbf{x}_0$ be any vector in $\mathbf{R}^N$ (not necessarily a codeword). Imagine starting at $\mathbf{x}_0$ and moving along a semi-infinite straight line (a *ray*) toward $\mathbf{x}$ (and beyond). One of two outcomes is possible:

---

[1] For any vector $\mathbf{x} \in \mathbf{R}^N$, $|\mathbf{x}| \equiv (\sum_i x_i^2)^{1/2}$.

either this line eventually crosses into some other decoding set, or else the entire line lies within D_0. Let P_0 be the set of points $\mathbf{x} \in \mathbf{R}^N - \{\mathbf{x}_0\}$ for which the entire ray remains in D_0, and let P_i comprise those points for which the ray crosses *first* from D_0 into D_i.

The set P_i is a *cone* with vertex at $\mathbf{x}_0$ (i.e. $\mathbf{x} \in P_i$ implies $\mathbf{x}_0 + \alpha(\mathbf{x}-\mathbf{x}_0) \in P_i$ for all $\alpha > 0$). Also, the boundary of P_i is formed by at most $M-2$ supporting hyperplanes (each passing through $\mathbf{x}_0$). Thus, P_i has polyhedral sections. Clearly, the M cones $P_0, \cdots, P_{M-1}$ are disjoint and partition $\mathbf{R}^N - \{\mathbf{x}_0\}$; consequently,

$$D_0^c = \bigcup_{i=1}^{M-1} (P_i \cap D_0^c) \tag{2.5}$$

since $D_0^c \cap P_0 = \emptyset$. From (2.3), we can write

$$P_i \cap D_0^c = P_i \cap \left(\bigcup_{j=1}^{M-1} H_i \right) = P_i \cap H_i \; . \tag{2.6}$$

The last equality follows from the definition of P_i: If we begin at $\mathbf{x}_0$ and proceed along any ray in P_i, we cross the boundary of H_i before that of H_j for all $j \neq i$; hence, $H_j \cap P_i \subset H_i \cap P_i$ for all $j \neq i$. Intuitively, $P_i \cap H_i$ is a truncated polyhedral cone with vertex at $\mathbf{x}_0$.

Combining (2.5) and (2.6), we obtain a disjoint partition of D_0^c:

$$P_{e,0} = Pr\,\{\mathbf{x}_0 + \mathbf{n} \in \bigcup_{i=1}^{M-1} (P_i \cap H_i)\,\} = \sum_{i=1}^{M-1} Pr\,\{\mathbf{x}_0 + \mathbf{n} \in P_i \cap H_i\,\}. \tag{2.7}$$

An Upper Bound on the Error: Since P_i is a cone we can define the solid angle as follows: Let Ω be the unit N-sphere centered at $\mathbf{x}_0$. The *solid angle* of P_i, denoted by $|\,P_i\,|$, is the surface area of the region $P_i \cap \Omega$ on Ω; similarly, $|\,\Omega\,|$ is the entire surface area of Ω.

The upper bound is obtained by deforming the polyhedral cones in (2.7) into circular cones with the same solid angle. Let $C_i(\theta)$ be a right-circular cone with half-angle θ, vertex at $\mathbf{x}_0$, and positive axis coincident with the ray from $\mathbf{x}_0$ through $\mathbf{x}_i$. Further, let $C_i = C_i(\theta_i)$, where θ_i is such that

$$|\,C_i(\theta_i)\,| = |\,P_i\,| \; . \tag{2.8}$$

We claim that

$$Pr\,\{\mathbf{x}_0 + \mathbf{n} \in P_i \cap H_i\,\} \le Pr\,\{\mathbf{x}_0 + \mathbf{n} \in C_i \cap H_i\,\} \; . \tag{2.9}$$

To prove (2.9), note that the noise vector $\mathbf{n}$ can be written as

$$\mathbf{n} = \sigma\, \chi_N\, \mathbf{w} \; , \tag{2.10}$$

where $\mathbf{w} \equiv \mathbf{n}/|\,\mathbf{n}\,|$ and $\chi_N \equiv |\,\mathbf{n}\,|$ are independent random variables. Here, χ_N^2 is a standard chi-squared random variable with N degrees of freedom and distribution function

$$Pr\,\{\,\chi_N \le x\,\} = \bar{\gamma}(N/2, x^2/2) \; , \tag{2.11}$$

where $\bar{\gamma}$ is as defined in (1.8), and $\mathbf{w}$ is uniformly distributed on the unit N-sphere, so that

$$Pr\,\{\,\mathbf{w} \in P\,\} = |\,P\,| \,/\, |\,\Omega\,| \tag{2.12}$$

for any cone P with vertex at the origin. We can write

$$\begin{aligned} Pr\,\{\,\mathbf{x}_0 + \mathbf{n} \in P_i \cap H_i\,\} &= Pr\,\{\,\mathbf{x}_0 + \mathbf{n} \in P_i \cap C_i \cap H_i\,\} \\ &\quad + Pr\,\{\,\mathbf{x}_0 + \mathbf{n} \in P_i \cap C_i^c \cap H_i\,\} \; . \end{aligned} \tag{2.13}$$

Let $d_i \equiv |\,\mathbf{x}_i - \mathbf{x}_0\,|$. Note that if $\mathbf{x}_0 + \mathbf{n}$ lies outside of $C_i(\theta)$ then $|\,\mathbf{n}\,| > (d_i/2)\sec\theta$ is *necessary* for $\mathbf{x}_0 + \mathbf{n}$ to lie in H_i, whereas if $\mathbf{x}_0 + \mathbf{n} \in C_i(\theta)$ then $|\mathbf{n}| > (d_i/2)\sec\theta$ is *sufficient* for $\mathbf{x}_0 + \mathbf{n}$ to lie in H_i. Hence

$$Pr\,\{\mathbf{x}_0 + \mathbf{n} \in P_i \bigcap C_i^c \bigcap H_i\,\} \le Pr\,\{\chi_N \ge (d_i/2)\sec\theta_i\,,\ \mathbf{x}_0 + \mathbf{w} \in P_i \bigcap C_i^c\,\} \tag{2.14}$$

$$Pr\,\{\mathbf{x}_0 + \mathbf{n} \in C_i \bigcap P_i^c \bigcap H_i\,\} \ge Pr\,\{\chi_N \ge (d_i/2)\sec\theta_i\,,\ \mathbf{x}_0 + \mathbf{w} \in C_i \bigcap P_i^c\,\}\,. \tag{2.15}$$

Since $|\,C_i\,| = |\,P_i\,|$, it follows that $|\,P_i \bigcap C_i^c\,| = |\,C_i \bigcap P_i^c\,|$; hence,

$$\begin{aligned} &Pr\,\{\chi_N \ge (d_i/2)\sec\theta_i\,,\ \mathbf{x}_0 + \mathbf{w} \in P_i \bigcap C_i^c\,\} \\ &= Pr\,\{\chi_N \ge (d_i/2)\sec\theta_i\,\}\ |\,P_i \bigcap C_i^c\,| \,/\, |\,\Omega\,| \\ &= Pr\,\{\chi_N \ge (d_i/2)\sec\theta_i\,\}\ |\,C_i \bigcap P_i^c\,| \,/\, |\,\Omega\,| \\ &= Pr\,\{\chi_N \ge (d_i/2)\sec\theta_i\,,\ \mathbf{x}_0 + \mathbf{w} \in C_i \bigcap P_i^c\,\}\,. \end{aligned} \tag{2.16}$$

Combining (2.14) - (2.16), we have proved

$$Pr\,\{\mathbf{x}_0 + \mathbf{n} \in P_i \bigcap C_i^c \bigcap H_i\,\} \le Pr\,\{\mathbf{x}_0 + \mathbf{n} \in C_i \bigcap P_i^c \bigcap H_i\,\}. \tag{2.17}$$

Substitution of (2.17) into (2.13) proves (2.9).

Next we derive an explicit formula for $Pr\,\{\mathbf{x}_0 + \mathbf{n} \in C_i \bigcap H_i\,\}$. The solid angle of a right-circular cone of half-angle θ is (e.g. [1, eq. 21])

$$|\,C_i(\theta)\,| = \frac{(N-1)\pi^{(N-1)/2}}{\Gamma((N+1)/2)} \int_0^{\theta} (\sin\phi)^{N-2}\,d\phi\,,\ 0 \le \theta < \pi, \tag{2.18}$$

and

$$|\,\Omega\,| = \frac{N\ \pi^{N/2}}{\Gamma(N/2+1)}\,. \tag{2.19}$$

Define $\mathbf{u}_i \equiv (\mathbf{x}_i - \mathbf{x}_0) / |\,\mathbf{x}_i - \mathbf{x}_0\,|$ and

$$\Phi_i \equiv \cos^{-1}(\mathbf{w} \cdot \mathbf{u}_i) \tag{2.20}$$

where $\cos^{-1}(x)$ has range $[0,\pi]$. Since $\mathbf{w}$ is independent of χ_N, Φ_i is also and

$$Pr\,\{0 \le \Phi_i \le \theta\} = |\,C_i(\theta)\,| \,/\, |\,\Omega\,|\,, \quad 0 \le \theta \le \pi\,; \tag{2.21}$$

thus, Φ_i has density

$$\begin{aligned} g(\phi) &= \frac{d}{d\phi}\,Pr\,\{0 \le \Phi_i \le \phi\} \\ &= \begin{cases} \dfrac{\Gamma(N/2)}{\sqrt{\pi}\,\Gamma((N-1)/2)}(\sin\phi)^{N-2} & 0 \le \theta \le \pi \\ 0 & \text{otherwise}\,. \end{cases} \end{aligned} \tag{2.22}$$

Using (2.11), we conclude that

$$\begin{aligned} Pr\,\{\mathbf{x}_0 + \mathbf{n} \in C_i \bigcap H_i\,\} &= Pr\,\{\,0 \le \Phi_i \le \theta_i\,,\ \mathbf{n}\cdot\mathbf{u}_i \ge |\,\mathbf{x}_i - \mathbf{x}_0\,|\,/2\,\} \\ &= Pr\,\{\,0 \le \Phi_i \le \theta_i\,,\ \chi_N\ \cos\Phi_i \ge d_i/2\sigma\,\} \\ &= \int_0^{\theta_i} Pr\,\{\,\chi_N \ge (d_i/2\sigma)\sec\phi\,\}\ g(\phi)\,d\phi \\ &= B_N(\theta_i\,,\ d_i/2\sigma) \end{aligned} \tag{2.23}$$

where

$$B_N(\theta,x) \equiv \frac{\Gamma(N/2)}{\sqrt{\pi}\Gamma((N-1)/2)} \int_0^{\theta} (\sin\phi)^{N-2}\, \overline{\Gamma}\left(\frac{N}{2}, \frac{x^2\sec^2\phi}{2}\right) d\phi. \tag{2.24}$$

It is not difficult to see that $B_N(\theta,x)$ represents the probability that $\tilde{\mathbf{n}}$, an i.i.d. $N(0,1)$ vector, lies in a cone of half-angle θ truncated at a distance x from the origin. A second useful expression for $B_N(\theta,x)$ is obtained by conditioning on the noise along the axis of the cone. Thus,

$$\begin{aligned} B_N(\theta,x) &= \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)\, Pr\,\{\chi_{N-1} \le z\tan\theta\}\, dz \\ &= \int_x^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)\, \overline{\gamma}\left(\frac{N-1}{2}, \frac{z^2\tan^2\theta}{2}\right) dz. \end{aligned} \tag{2.25}$$

Finally, observe that for $x=0$ (2.24) yields

$$\begin{aligned} B_N(\theta,0) &= \frac{\Gamma(N/2)}{\sqrt{\pi}\Gamma((N-1)/2)} \int_0^{\theta} (\sin\phi)^{N-2}\, d\phi \\ &= |\, C_i(\theta)\,|\, /\, |\,\Omega\,|\,. \end{aligned} \tag{2.26}$$

Thus, the condition (2.8) can also be expressed in terms of B_N.

Summarizing, we have proved the following proposition.

Proposition 1: Let $\mathbf{x}_0, \cdots, \mathbf{x}_{M-1}$ be an arbitrary code used with the maximum likelihood decoder in (2.1). Then

$$P_{e,0} \le \sum_{i=1}^{M-1} B_N(\,\theta_i\,,\, d_i/2\sigma\,) \tag{2.27}$$

where θ_i satisfies

$$B_N(\theta_i,0) = |\, P_i\,|\, /\, |\,\Omega\,|\,,\ 1 \le i \le M-1\,, \tag{2.28}$$

and $B_N(\theta,x)$ is given in (2.24) or (2.25).

●

Some insight into Proposition 1 can be gained by examining the properties of $B_N(\theta,x)$. B_N is increasing in θ and decreasing in x. Also $B_N(\pi/2,x) = Q(x)$, so the bound is always tighter than the union bound.

It is generally very difficult to determine the solid angles $|\,P_0\,|, \cdots, |\,P_{M-1}\,|$. However, in some cases they can be obtained by symmetry. We give only one example: equal-energy orthogonal signals.

Example: Equal-Energy Orthogonal Signals. Without loss of generality, let $\mathbf{x}_0 = \sqrt{E}\,\mathbf{e}_0, \cdots, \mathbf{x}_{M-1} = \sqrt{E}\,\mathbf{e}_{M-1}$, where $\mathbf{e}_0, \cdots, \mathbf{e}_{M-1}$ is the standard basis for $\mathbf{R}_N$. It is not difficult to verify that

$$P_i = \{\mathbf{x} \in \mathbf{R}_N \;\; | \;\; |\,x_i\,| \ge \max_{j \ne i} |\,x_j\,|\,\}\,,\ 0 \le i \le M-1, \tag{2.29}$$

except, perhaps, on the boundary. By symmetry $|\,P_i\,| = |\,\Omega\,|/M$, and hence $\theta_i = \theta_0$ for all i, where θ_0 satisfies

$$B_M(\theta_0,0) = \frac{1}{M}\,, \tag{2.30}$$

and the bound in Proposition 1 becomes

$$P_{e,0} \leq (M - 1)\, B_N(\, \theta_o \,, \sqrt{E/2\sigma^2})\,. \tag{2.31}$$

This bound can be made slightly tighter by observing that subtracting $\mathbf{x} = (1/M) \sum \mathbf{x}_i$ from each signal converts the signal set to an $(M-1)$-dimensional *simplex* signal set with energy $E' = (M-1)E/M$, without altering performance. Thus, we can replace B_M in (2.30) and (2.31) with B_{M-1} and E with E'. Note that θ_0 does not depend upon signal-to-noise ratio. Also, any θ for which $M \; B_{M-1}(\theta,0) \geq 1$ gives an upper bound in (2.31).

A General Upper Bound: The symmetry arguments leading to (2.30)-(2.31) clearly do not extend to general signal sets, so that the solid angles $| P_0 |, \ldots, | P_{M-1} |$ are generally unavailable. We are interested in obtaining an upper bound applicable to any set of codewords. Such a bound is provided by

$$P_{e,0} \leq \max_{\theta_1, \cdots, \theta_{M-1}} \sum_{i=1}^{M-1} B_N(\, \theta_i \,, d_i/2\sigma\,) \tag{2.32}$$

where the maximum is taken over all $\theta_1, \cdots, \theta_{M-1}$ such that

$$\sum_{i=1}^{M-1} B_N(\theta_i, 0) \leq 1\,. \tag{2.33}$$

First consider the special case of this optimization problem for $M=3$. Here, we wish to maximize

$$B_N(\theta_1, d_1/2\sigma) + B_N(\theta_2, d_2/2\sigma) \tag{2.34}$$

subject to the constraint

$$B_N(\theta_1,0) + B_N(\theta_2,0) = \beta \leq 1\,. \tag{2.35}$$

Note that $B_N(\theta, x)$ can be viewed as an integral of conical elements with vertices at $\mathbf{x}_0$. The contribution of a conical element at an angle ϕ from the axis of the cone is decreasing with the length of the element $x \sec \phi$. Note that a conical element at the periphery of the first cone is truncated at a distance of $(d_1/2)\sec\theta_1$ from the vertex. Thus if $(d_1/2)\sec\theta_1 > (d_2/2)\sec\theta_2$, we can subtract solid angle (decrease θ_1) from the periphery of the first cone, and add an equal amount to the second cone (increase θ_2 to get equality in (2.35)), thereby increasing the sum (2.34). We can continue this until the angles of the cones satisfy $\theta_1 = 0$ or $d_1 \sec\theta_1 = d \sec\theta_2$, at which point further change reduces the value of (2.34).

In a similar way, it is straightforward to show that the maximum in (2.34) occurs when

$$d_i \;\sec\theta_i = \alpha\,, \text{ for all } \theta_i > 0\,, \tag{2.36}$$

for some $\alpha > 0$. This gives rise to a one-parameter bound: Define

$$\Theta(y) \equiv \begin{cases} \cos^{-1}(y) & | y | \leq 1 \\ 0 & | y | > 1 \end{cases}. \tag{2.37}$$

Then for any codevectors

$$P_{e,0} \leq \sum_{i=1}^{M-1} B_N(\, \Theta(d_i/\alpha)\,, d_i/2\sigma\,) \tag{2.38}$$

where α is such that

$$\sum_{i=1}^{M-1} B_N(\Theta(d_i/\alpha), 0) = 1\,. \tag{2.39}$$

Note that the optimizing value of α in (2.39) does not depend upon the noise variance σ^2. Further, we note that

$$h(\alpha) \equiv \sum_{i=1}^{M-1} B_N(\Theta(d_i/\alpha), 0) \tag{2.40}$$

is increasing; $h(\alpha) = 0$ for $\alpha \le \min_i d_i$ and for α greater than this value, $h(\alpha)$ is strictly increasing to $(M-1)/2$ at $\alpha = +\infty$. Thus there is a unique α_o that gives equality in (2.40).

Thus we have proved the following proposition which is the main result of this paper.

Proposition 2: Let $\mathbf{x}_0, \cdots, \mathbf{x}_{M-1}$, be any code used with minimum distance decoding. Then

$$P_{e,0} \le \sum_{i=1}^{M-1} A_N(d_i/\alpha^o, d_i/2\sigma) \tag{2.41}$$

where $d_i \equiv |\mathbf{x}_i - \mathbf{x}_0|$ and α_o is the unique solution of

$$\sum_{i=1}^{M-1} A_N(d_i/\alpha^o, 0) = 1 \,. \tag{2.42}$$

and

$$A_N(y, x) \equiv B_N(\Theta(y), x) \,. \tag{2.43}$$

•

It is not difficult to show that (2.41) is always tighter than both bounds (1.2) and (1.3). However, the bound only gives significantly tighter results than the union bound for error probabilities of interest when $M >> N$. In particular, the bound is rather poor for orthogonal signals.

3. Expansions

In this section, we derive several series expansions for $B_N(\theta, x)$ that can be used to calculate the bound in Proposition 2. These expansions involve the incomplete gamma functions which can be calculated using standard algorithms.

A Recursion for $B_N(\theta,x)$: For all a and x, the incomplete gamma function satisfies the recursion [2, Section 9.2, eq. 2]

$$\gamma(a+1, x) = a\,\gamma(a, x) + x^a e^{-x} \,, \tag{3.1}$$

where $\gamma(a,x) = \Gamma(a)\overline{\gamma}(a,x)$. Substitution of (3.1) with $a = (N-3)/2$ into (2.25) yields after some simplification

$$B_N(\theta, x) = B_{N-2}(\theta, x) + \frac{\cos\theta(\sin\theta)^{N-3}}{2\sqrt{\pi}\,\Gamma((N-1)/2)}\,\Gamma\left(\frac{N-2}{2}, \frac{x^2\sec^2\theta}{2}\right) \tag{3.2}$$

where $\Gamma(a,x) = \Gamma(a)\overline{\Gamma}(a,x)$. Applying (3.2) to even and odd N, we obtain the following expansions for $B_N(\theta, x)$:

$$B_{2K}(\theta, x) = B_2(\theta, x) - \frac{\cos\theta}{2\sqrt{\pi}} \sum_{m=1}^{K-1} \frac{(\sin\theta)^{2m-1}}{\Gamma(m+\frac{1}{2})}\,\Gamma\left(m, \frac{x^2\sec^2\theta}{2}\right), \tag{3.3a}$$

$$B_{2K+1}(\theta, x) = B_3(\theta, x) - \frac{\cos\theta}{2\sqrt{\pi}} \sum_{m=1}^{K-1} \frac{(\sin\theta)^{2m}}{\Gamma(m+1)}\,\Gamma\left(m+\frac{1}{2}, \frac{x^2\sec^2\theta}{2}\right). \tag{3.3b}$$

We make two observations about (3.3). First, when the argument a is a positive integer, as in (3.3a), then $\Gamma(a,x)$ is given by the simple expression [2, Section 9.2, eq. 18]

$$\Gamma(a, x) = (a-1)!\,e^{-x} \sum_{k=0}^{a-1} \frac{x^k}{k!} \,; \tag{3.4}$$

when $a = m+½$, for m an integer, the recursion (3.1) gives a series for $\Gamma(a,x)$ involving a sum of terms of the form $x^a \, e^{-x}$ and $\Gamma(1/2, x^2\sec^2\theta/2)$. Here we can use the identity [2, Section 9.9, eq. 2]

$$\Gamma(1/2, z^2/2) = 2\sqrt{\pi}Q(z), \tag{3.5}$$

so that the incomplete gamma functions in (3.3) can be calculated using only elementary functions and $Q(z)$.

In (3.3), it remains only to calculate $B_2(\theta,x)$ and $B_3(\theta,x)$. From (3.4), it follows that $\bar{\gamma}(1,x) = 1 - e^{-x}$; hence, from (2.25), we find that

$$B_3(\theta,x) = Q(x) - \cos\theta \, Q(x\sec\theta). \tag{3.6}$$

It is more difficult to evaluate $B_2(\theta,x)$. From Proposition 2, we can always use $N+1$ in the bound in place of N, so that (3.3b) can always be used for a bound. However, the tightest bound is obtained when N is as small as possible. From (2.25) and (3.5), we obtain

$$B_2(\theta,x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp(-z^2/2) \left[1 - 2Q(z^2\tan^2\theta/2) \right] dz. \tag{3.8}$$

This integral can always be evaluated numerically, especially if θ is small. Here, it is useful to note that

$$B_2(\theta,x) + B_2(\pi/2-\theta, x\tan\theta) = Q(x) + Q(x\tan\theta) - 2Q(x)\,Q(x\tan\theta), \tag{3.9}$$

so that (3.8) can always be evaluated for $\theta < \pi/4$. Alternatively, we can write

$$B_2(\theta,x) = \frac{\theta}{\pi} - \frac{2\cot\theta}{\pi} G\left(\frac{x\tan\theta}{\sqrt{2}}, \cot\theta \right), \tag{3.10}$$

where

$$G(z,p) \equiv \int_0^z e^{-p^2y^2} \int_0^y e^{-x^2} \, dx dy. \tag{3.11}$$

The function $G(z,p)$ arises in rocketry, and has been studied extensively by Rosser [3], who gives 18 different series expansions for computing it.

Finally, we note that a simple, finite series for $B_N(\theta,0)$ can be obtained by substituting the following identities from [4, Section 2.513] into (2.26):

$$\int_0^\theta \sin^{2K+1}\phi \, d\phi = \frac{(-1)^{K+1}}{4^K} \sum_{i=0}^{K} (-1)^i \binom{2K+1}{i} \frac{\cos(2K+1-2i)\theta}{2K+1-2i}, \tag{3.12a}$$

$$\int_0^\theta \sin^{2K}\phi \, d\phi = \binom{2K}{K}\frac{\theta}{4^K} + \frac{2(-1)^K}{4^K} \sum_{i=0}^{K} (-1)^i \binom{2K}{K} \frac{\sin(2K-2i)\theta}{2K-2i}, \tag{3.12b}$$

4. An Example

As an example, we have computed the bound of Proposition 2 for the Leech lattice code (eg. [5, pg. 131]). This code has blocklength $N=24$, each codevector has $N_{\min}=196560$ nearest neighbors, and all other codewords are at a distance of at least $3d_{\min}/2$ from $\mathbf{x}_0$. Here, the optimal angles are $\theta_i = \theta_o \equiv 0.7065$ radians for all codevectors at minimum distance from $\mathbf{x}_0$, and $\theta_i = 0$ for all other codevectors. The bound is therefore

$$P_{e,0} \leq N_{\min} \, B_{24}(\theta_o, d_{\min}/2\sigma). \tag{4.1}$$

This bound is plotted in Figure 1. Also shown for comparison are the minimum distance bound

(1.3) and an approximation to the union bound $N_{\min} Q(d_{\min}/2\sigma)$. As expected, the minimum distance bound is very poor for this large value of N. The bound of Proposition 2 improves upon the union approximation by roughly half a dB at an error probability of 0.1, and a quarter dB at 0.01. These improvements are fairly typical of lattice codes where $M >> N$. For less densely packed code the improvements will be smaller.

References

1. C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611-656, May 1959.
2. A. Erdelyi, *Higher Trancendental Functions, vol. II.* New York: McGraw-Hill, 1953.
3. J. B. Rosser, *Theory and Application of* $\int_0^z \exp(-x^2)dx$ and $\int_0^z \exp(-p^2y^2)\int_0^y exp(-x^2)dxdy$. New York: Mapleton House, 1948.
4. I. S. Gradshteyn and I. M. Ryzhik, *Tables of Integrals, Series and Products.* New York: Academic Press, 1980.

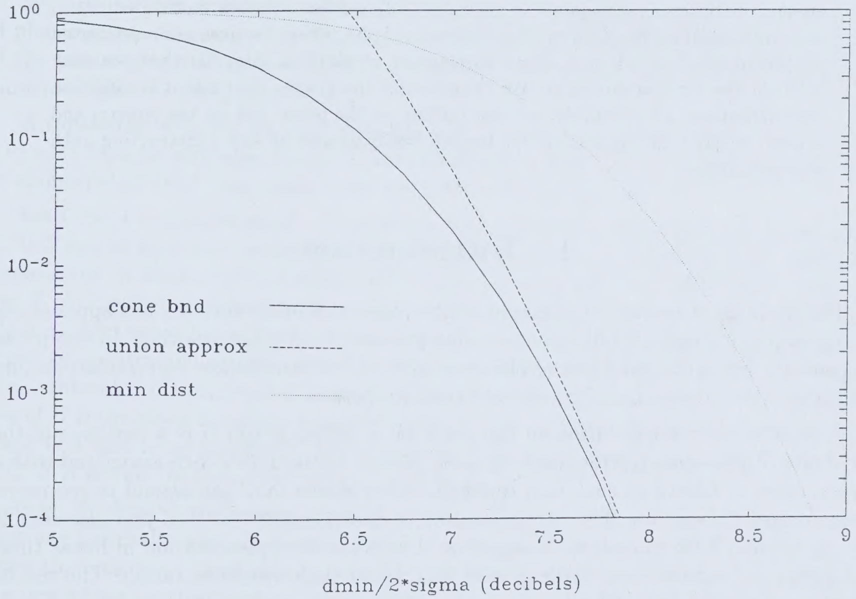


Figure 1. A comparison of bounds for the Leech lattice code

Tessellation Representations of Planar Graphs[*]

Roberto Tamassia
Department of Computer Science
Brown University
Providence, RI 02912-1910

Ioannis G. Tollis
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0668

Abstract

We investigate representations of graphs on the plane and on the sphere by means of the geometric relations of adjacency and visibility with respect to two families of orthogonal lines. We study *tessellation representations* of planar graphs where each constituent (vertex, edge, and face) of an embedded planar graph is represented by a rectangle with horizontal and vertical sides, and incidencies between constituents correspond to geometric adjacencies between rectangles. Also, we study *visibility representations* of planar graphs, where vertices are represented by horizontal intervals and edges correspond to pairs of intervals that see each other in the vertical direction. We characterize the graphs that admit tessellation representations and visibility representations on the plane and on the sphere, and present linear-time algorithms for testing the existence of and constructing such representations.

1 Introduction

The problem of representing graphs in the plane and other surfaces has applications in many areas including VLSI layout, motion planning, and graph drawing. In this paper we consider two representations of planar graphs, called *tessellation representations* and *visibility representations*, and investigate their properties.

A *tessellation representation* on the plane for a planar graph G is a partition of the plane into disjoint *tiles* (rectangles with sides parallel to the axes), each associated with a vertex, edge, or face of G, such that topological incidencies in G correspond to geometric adjacencies between tiles. This representation is closely related to VLSI *floorplanning* [9, 19]. In Section 3 we show how to construct a tessellation representation in linear time, and give a characterization of the graphs that admit such a representation. The results are also extended to the sphere.

The concept of visibility plays a fundamental role in a variety of applications [8]. Let I be a set of horizontal intervals (segments with or without either endpoint) in the plane. The *visibility graph* of I is the graph whose vertices are the intervals of I, and whose edges connect pairs of intervals that can see each other in the vertical direction. Conversely, a *visibility representation* on the plane for a graph G is a set of parallel intervals in the plane whose visibility graph is isomorphic to G. The concepts of visibility graph and visibility representation can be naturally extended to other surfaces where two families of parallel lines are defined. Visibility representations find specific application to VLSI layout compaction [6,12], motion planning [3,10], and graph drawing [1,16].

---

[*] Research partially supported by Cadre Technologies, Inc., and by the Texas Advanced Research Program under grant no. 3972.

Many authors have investigated visibility representations on the plane [2,5,11,14,15, 17,18,20,21]. The classes of graphs that admit a visibility representation on the plane and on the cylinder have been characterized in [14,20] and [15,17], respectively. Linear-time algorithms for testing the existence of and constructing such representations are given in [14,15,17].

In Section 4 we consider the problem of constructing visibility representations on the sphere, where the intervals are subsets of parallels, and visibility is along meridians. We characterize the class of graphs that admit a visibility representation on the sphere and give linear time algorithms for testing the existence of and constructing such a representation. The related problem of characterizing *spherical orders* is investigated in [3].

2 Planar Digraphs and Orthogonal Intervals

We assume that the reader is familiar with graph terminology and basic properties of planar graphs (see, e.g., [7]). Throughout this paper n denotes the number of vertices of the planar graph G currently being considered. Recall that a planar graph without multiple edges has $O(n)$ edges.

A *topological ordering* for an acyclic digraph G maps each vertex v of G to an integer $\xi(v)$ such that for any edge (u, v) of G we have $\xi(u) < \xi(v)$. A topological ordering can be computed in $O(n)$ time using a variation of depth-first-search.

Let G be a connected graph. A *cutvertex* of G is a vertex whose removal disconnects G. If G has at least one cutvertex, it is said to be 1-*connected*, otherwise it is said to be 2-*connected*. A *block* of G is a maximal 2-connected subgraph of G. The *block-cutvertex* tree T of G is the tree whose vertices are the blocks and the cutvertices of G, and whose edges connect every block B to the cutvertices contained in B. The block-cutvertex tree T can be constructed in $O(n)$ time by using a variation of depth-first search. A *cutface* of an embedded planar graph G is a cutvertex of the dual graph of G. The *block-cutface* tree of G is the block-cutvertex tree of the dual graph. A block B of a planar 1-connected graph G is said to be *k-covered* if there is an embedding of B such that all the cutvertices of G in B are on the boundary of at most k faces.

Now, we define some special types of trees (see Figure 1). A *star* is a tree that has at most one internal vertex. A *chain* is a tree that has at most two leaves. A *double-star* is a tree that consists of two stars connected by a chain of length at most two. A *centipede* is a tree whose non-leaf vertices form a chain. Notice that each such tree can degenerate to a single vertex.

In the following, V, E, and F denote the set of vertices, edges, and faces of a planar embedded digraph G, respectively. An element $c \in V \cup E \cup F$ is called a *constituent* of G. For each edge e, we define $low(e)$ and $high(e)$ as the tail and head vertices of e. Also, we define $left(e)$ to be the face to the left of e and $right(e)$ to be the face to the right of e, when e is traversed from $low(e)$ to $high(e)$.

A *spherical st-graph* is an embedded planar acyclic digraph with exactly one source (vertex without incoming edges), s, and exactly one sink (vertex without outgoing edges), t. This definition differs from the one originally given in [13] because it requires acyclicity. A *planar st-graph* is a spherical st-graph such that s and t are on the boundary of the external face. It is convenient to visualize a spherical st-graph as drawn on a sphere with s at the "South Pole" and t at the "North Pole". Analogously, we visualize a planar st-graph as drawn in the plane with s at the bottom and t at the top. Spherical st-graphs were demonstrated to have the following important properties [4,13,14]:

1. Every vertex is on a directed path from s to t.
2. The incoming edges of each vertex v appear consecutively around v, and so do the outgoing edges. The face separating the incoming and outgoing edges of v in the clockwise direction is called $\mathit{left}(v)$, and the face separating them in the counterclockwise direction is called $\mathit{right}(v)$ (see Figure 2).
3. The boundary of each face f consists of two directed paths enclosing f. The first and last vertices of these paths are called $\mathit{low}(f)$ and $\mathit{high}(f)$, respectively (see Figure 2).

The terminology is extended by defining vertices $\mathit{low}(c)$ and $\mathit{high}(c)$ and faces $\mathit{left}(c)$ and $\mathit{right}(c)$ for all constituents. For each vertex v, we define $\mathit{low}(v) = \mathit{high}(v) = v$ and $\mathit{left}(v)$ and $\mathit{right}(v)$ as above. For each face f, we define $\mathit{low}(f)$ and $\mathit{high}(f)$ as above and $\mathit{left}(f) = \mathit{right}(f) = f$.

The *dual* graph G^* of a planar st-graph G is the planar st-graph obtained as follows (see Figure 3):

1. construct the digraph that has a dual vertex for each face of G, and a dual edge e^* for each edge e of G, directed from $\mathit{left}(e)$ to $\mathit{right}(e)$;
2. replace the dual vertex f associated with the external face with two new vertices, denoted s^* and t^*, where s^* inherits the outgoing edges, and t^* inherits the incoming edges of f.

Now, we give some geometric definitions. A *cylinder* C is the locus of points at the same distance from a straight line, called the *axis* of the cylinder. The notion of horizontal and vertical is extended to the cylinder by considering horizontal the lines parallel to the axis, and vertical the circles on a plane orthogonal to the axis.

A *sphere* S is the locus of points at the same distance from a point, called the *center* of the sphere. The intersection of S with the horizontal plane that passes through the center of S defines a circle, called the *equator*. Similarly, the intersection of S with planes parallel to the plane of the equator defines the *parallels*. The line that passes through the center of S and is orthogonal to the plane of the equator, called the *axis* of the sphere, intersects the sphere into two points, the *North Pole* and the *South Pole*. Every plane that is orthogonal to the plane of the equator and passes through the two Poles defines a circle, called a *meridian*. Every point p of S will be denoted by a pair (α, β), where α is the latitude measured with respect to the South Pole, and β is the longitude measured with respect to a reference meridian. The notion of horizontal and vertical is extended to the sphere by considering horizontal the parallels and vertical the meridians.

3 Tessellation Representations

A *tile on the plane* is a rectangle delimited by two horizontal lines and two vertical lines. A *tile on the sphere* is the portion of the sphere delimited by two parallels and two meridians. On the plane, we allow unbounded tiles and tiles at infinity. On the sphere, we allow tiles containing one or both Poles.

Let G be an embedded planar digraph, and S denote either the plane or the sphere. A *tessellation representation* Θ for G on S maps each constituent (element of $V \cup E \cup F$) c of G into a tile $\Theta(c)$ of S such that:

1. The interiors of tiles $\Theta(c)$ and $\Theta(d)$ are disjoint whenever $c \neq d$.
2. The union of all tiles $\Theta(c)$, $c \in V \cup E \cup F$, is equal to S.
3. For each edge e, the bottom (resp. top, left, right) side of tile $\Theta(e)$, lies on the top (resp. bottom, right, left) side of tile $\Theta(low(e))$ (resp. $\Theta(high(e))$, $\Theta(left(e))$, $\Theta(right(e))$).

The following algorithm, which is a variation of the ones for visibility representations given in [1,14], constructs a tessellation representation on the plane for a planar *st*-graph G.

1. Compute a topological ordering ξ of the vertices of G, with $\xi(s) = 0$.
2. Construct the dual graph G^* of G, and compute a topological ordering ξ^* of the vertices of G^*, with $\xi^*(s^*) = 0$.
3. Define $\xi^*(f) = \infty$ for the external face f.
4. Let $\Theta(s)$ and $\Theta(t)$ be the unbounded tiles defined by $y \leq \xi(s)$ and $y \geq \xi(t)$, respectively. Also, for the external face f, let $\Theta(f)$ be the degenerate tile consisting of the line at infinity.
5. For each c distinct from s, t, and f, draw $\Theta(c)$ with coordinates (see Figure 4):

$$x_1(c) = \xi^*(left(c)); \quad x_2(c) = \xi^*(right(c)); \quad y_1(c) = \xi(low(c)); \quad y_2(c) = \xi(high(c)).$$

Notice that in the tessellation representation constructed by the above algorithm the tiles associated with the faces and the vertices (except s and t) are degenerate. In particular, the external face is associated with the a tile at infinity. It is possible to modify the construction so that only one tile is degenerate, namely, the one associated with the external face.

In Figure 3 the vertices of G and G^* are labeled with the values of topological orderings ξ and ξ^*, respectively. Figure 5 shows the tessellation representation on the plane computed by our algorithm for the planar *st*-graph G of Figure 3.

Theorem 1 *An embedded planar digraph G with n vertices admits a tessellation representation on the plane if and only if it is a planar st-graph. Also, a tessellation representation for G on the plane can be constructed in $O(n)$ time.*

Now, we consider a spherical *st*-graph G. Let π be a path in G from s to t, where we assume that π does not contain its extreme vertices s and t. We construct from G a new planar *st*-graph G_π by "cutting" G along path π and duplicating the vertices and edges of π. Note that the graph G_π is a planar *st*-graph. The two copies of π, denoted π' and π'' are the left and right boundary of G_π, respectively. Also, we denote by c' and c'' the two copies of a vertex or edge c of π in π' and π'', respectively. For any other constituent

c of G which is not in π, both c' and c'' denote the unique constituent of G_π associated with c.

The following algorithm constructs a tessellation representation on the sphere for a spherical *st*-graph G. It essentially constructs a tessellation representation on the plane for G_π with the property that for each constituent c of π the tiles $\Theta(c')$ and $\Theta(c'')$ have the same y-coordinates. This representation is subsequently "wrapped around" the sphere and the tiles $\Theta(c')$ and $\Theta(c'')$ are joined together.

1. Compute a topological ordering ξ of the vertices of G, with $\xi(s) = 0$
2. Find a path π from s to t and construct the associated planar *st*-graph G_π.
3. Construct the dual graph G^*_π of G_π and compute a topological ordering ξ^* of the vertices of G^*_π, with $\xi^*(s^*) = 0$.
4. Set $\alpha_0 = \pi/\xi(t)$ and $\beta_0 = 2\pi/\xi^*(t^*)$.
5. Let $\Theta(s)$ and $\Theta(t)$ be the degenerate tiles at the South Pole and North Pole, respectively.
6. For each c distinct from s and t, draw $\Theta(c)$ with coordinates:

$$\alpha_1(c) = \xi(low(c))\alpha_0; \quad \alpha_2(c) = \xi(high(c))\alpha_0;$$
$$\beta_1(c) = \xi^*(left(c''))\beta_0; \quad \beta_2(c) = \xi^*(right(c'))\beta_0.$$

Notice that in the tessellation representation constructed by the above algorithm the tiles associated with the faces and the vertices are degenerate. It is possible to modify the construction so that no tile is degenerate.

Theorem 2 *An embedded planar digraph G with n vertices admits a tessellation representation on the sphere if and only if it is a spherical st-graph. Also, a tessellation representation for G on the sphere can be constructed in $O(n)$ time.*

Let G be a planar undirected graph. Graph G is said to be *spherically st-orientable* (resp. *planarly st-orientable*) if it can be oriented and embedded so that the resulting digraph is a spherical *st*-graph (resp. planar *st*-graph). It can be shown that G is spherically *st*-orientable if and only if it is possible to add an edge (s,t) to G such that the resulting graph is 2-connected. Also, G is planarly *st*-orientable if and only if it is possible to add an edge (s,t) to G such that the resulting graph is 2-connected and planar [4].

A tessellation representation for a planar undirected graph G is defined as a tessellation representation for some embedded orientation of G. We have:

Theorem 3 *Let G be a planar undirected graph with n vertices.*

1. *G admits a tessellation representation on the sphere if and only if it is possible to add an edge (s,t) to G such that the resulting graph is 2-connected.*
2. *G admits a tessellation representation on the plane if and only if it is possible to add an edge (s,t) to G such that the resulting graph is 2-connected and planar.*

Also, there are $O(n)$-time algorithms for testing the existence of and constructing a tessellation representation for G on the sphere or on the plane.

4 Visibility Representations

We define an *interval on the plane* as a topologically connected subset of a horizontal or vertical line in the plane. Analogously, we define an *interval on the sphere* as a topologically connected subset of a parallel or a meridian. An *interval on the cylinder* is a topologically connected subset of a horizontal line or of a vertical circle. The notion of interval allows either endpoint to be contained or not. On the plane or cylinder, an interval might be unbounded. On the sphere, the North and South Pole are special degenerate intervals. On the cylinder, circles are special degenerate intervals.

Let H be a set of horizontal intervals on the plane, sphere, or cylinder. Intervals η_1 and η_2 of H are said to be *visible* if there is a vertical interval that intersects η_1 and η_2 but no other interval of H. The *visibility graph* of H is the graph whose vertices are the intervals of H and whose edges are the pairs of visible intervals. A *visibility representation* (on the plane, sphere, or cylinder) for an undirected graph G is a set of intervals H (on the plane, sphere, or cylinder) whose visibility graph is isomorphic to G. Clearly, a graph that admits a visibility representation must be planar.

Previous results have led to the characterization of the graphs that admit visibility representations on the plane and on the cylinder. Let G be a planar undirected graph with n vertices. For visibility representations on the plane, the following conditions are equivalent [14,20]:

1. G admits a visibility representation on the plane.
2. Every block of G is 1-covered.
3. G admits an embedding whose block-cutface tree is a star.

For visibility representations on the cylinder, the following conditions are equivalent [15, 17]:

1. G admits a visibility representation on the cylinder.
2. The block-cutvertex tree of G is a centipede.
3. G admits an embedding whose block-cutface tree is a chain.

Also, there are $O(n)$-time algorithms for testing the existence of and constructing a visibility representation for G on the plane or on the cylinder.

In the rest of this section we consider the problem of characterizing the class of graphs that admit a visibility representation on the sphere. Clearly this class contains as a subclass the planar graphs that admit a visibility representation on the plane.

Theorem 4 *Let G be a planar undirected graph with n vertices. The following conditions are equivalent:*

1. G admits a visibility representation on the sphere.

2. There exists a spherical st-graph G' such that G is isomorphic to the undirected graph obtained from G' by ignoring the orientation of the edges and removing vertices s and t.

3. One of the following is true:

(a) At most one block of G is 2-covered and the other blocks are 1-covered.

(b) Two blocks of G, denoted B' and B'', are 2-covered and such that the path of the block-cutvertex tree T of G between B' and B'' contains 2-covered blocks, each having exactly two cutvertices. Also, blocks B' and B'' become 1-covered if we do not consider the cutvertices on the path of T from B' to B''. Finally, the remaining blocks are 1-covered.

Also, there are $O(n)$-time algorithms for testing the existence of and constructing a visibility representation on the sphere for G.

A simpler characterization is obtained if we do not allow horizontal intervals on the sphere that are complete parallels. We refer to this variation of visibility representation on the sphere as *open visibility representation* on the sphere.

Theorem 5 *Let G be a planar undirected graph with n vertices. The following conditions are equivalent:*

1. *G admits an open visibility representation on the sphere.*
2. *There exists a 2-connected spherical st-graph G' such that G is isomorphic to the undirected graph obtained from G' by ignoring the orientation of the edges and removing vertices s and t.*
3. *G admits an embedding such that all the cutvertices are on the boundary of at most two faces.*
4. *At most one block of G is 2-covered and the remaining blocks are 1-covered.*
5. *G admits an embedding whose block-cutface tree is a double-star.*

Also, there are $O(n)$-time algorithms for testing the existence of and constructing an open visibility representation on the sphere for G.

The relation between the various classes of visibility graphs is illustrated in Figure 6.

Acknowledgment

We would like to thank Jorge Urrutia and Ivan Rival for useful discussions.

References

[1] G. Di Battista & R. Tamassia, "Algorithms for Plane Representations of Acyclic Digraphs," *Theoretical Computer Science* 61 (1988), 175–198.

[2] P. Duchet, Y. Hamidoune, M. Las Vergnas & H. Meyniel, "Representing a Planar Graph by Vertical Lines Joining Different Levels," *Discrete Mathematics* 46 (1983), 319–321.

[3] S. Foldes, I. Rival & J. Urrutia, "Light Sources, Obstructions, and Spherical Orders," Dept. of Computer Science, Univ. of Ottawa, Technical Report TR-88-23, 1988.

[4] A. Lempel, S. Even & I. Cederbaum, "An Algorithm for Planarity Testing of Graphs," *Theory of Graphs, Int. Symposium* (1966), 215–232.

[5] F. Luccio, S. Mazzone & C. Wong, "A Note on Visibility Graphs," *Discrete Mathematics* 63 (1987), 105–110.

[6] K. Mehlhorn & W. Rulling, "Compaction on the Torus," *VLSI Algorithms and Architectures (Proc. AWOC '88, Corfu, Greece, 1988)* (1988), 212–225.

[7] T. Nishizeki & N. Chiba, *Planar Graphs: Theory and Algorithms*, Annals of Discrete Mathematics 32, North-Holland, 1988.

[8] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[9] R.H.J.M. Otten & J.G. van Wijk, "Graph Representations in Interactive Layout Design," *Proc. IEEE Int. Symp. on Circuits and Systems* (1978), 914–918.

[10] I. Rival & J. Urrutia, "Representing Orders by Translating Convex Figures in the Plane," *Order* 4 (1988), 319–339.

[11] P. Rosenstiehl & R.E. Tarjan, "Rectilinear Planar Layouts of Planar Graphs and Bipolar Orientations," *Discrete & Computational Geometry* 1 (1986), 342–351.

[12] M. Schlag, F. Luccio, P. Maestrini, D.T. Lee & C.K. Wong, "A Visibility Problem in VLSI Layout Compaction," in *Advances in Computing Research, vol. 2*, F.P. Preparata, ed., JAI Press Inc., Greenwich, CT, 1985, 259–282.

[13] R. Tamassia, "A Dynamic Data Structure for Planar Graph Embedding," *Automata, Languages and Programming (Proc. 15th ICALP, Tampere, Finland)* (1988), 576–590.

[14] R. Tamassia & I.G. Tollis, "A Unified Approach to Visibility Representations of Planar Graphs," *Discrete & Computational Geometry* 1 (1986), 321–341.

[15] R. Tamassia & I.G. Tollis, "Centipede Graphs and Visibility on a Cylinder," in *Graph-Theoretic Concepts in Computer Science, (Proc. Int. Workshop WG '86, Bernierd, June 1986)*, G. Tinhofer and G. Schmidt, ed., Lecture Notes in Computer Science, vol. 246, Springer-Verlag, 1987, 252–263.

[16] R. Tamassia & I.G. Tollis, "Planar Grid Embedding in Linear Time," *IEEE Trans. on Circuits and Systems* CAS-36 (1989), 1230–1234.

[17] R. Tamassia & I.G. Tollis, "Representations of Graphs on a Cylinder," Dept. of Computer Science, Brown Univ., Technical Report CS-89-10, 1989.

[18] C. Thomassen, "Plane Representations of Graphs," in *Progress in Graph Theory*, J.A. Bondy and U.S.R. Murty, ed., Academic Press, 1984, 43–69.

[19] S. Wimer, I. Koren & I. Cederbaum, "Floorplans, Planar Graphs, and Layouts," *IEEE Trans. on Circuits and Systems* 35 (1988), 267–278.

[20] S.K. Wismath, "Characterizing Bar Line-of-Sight Graphs," *Proc. ACM Symp. on Computational Geometry* (1985), 147–152.

[21] S.K. Wismath, "Weighted Visibility Graphs of Bars and Related Flow Problems," *Algorithms and Data Structures (Proc. WADS'89)* (1989), 325–334.

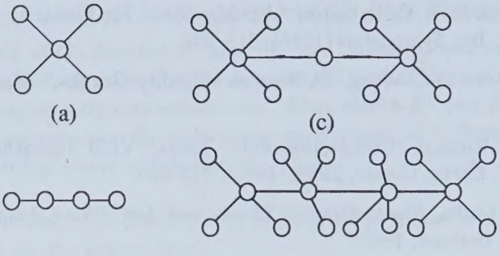


Figure 1. Examples of trees: (a) star; (b) chain; (c) double star; (d) centipede.

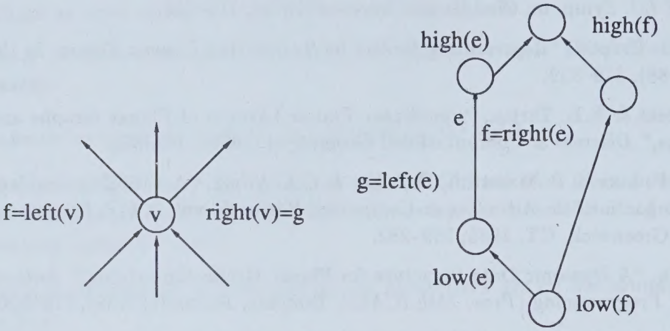

high(e)
high(f)
e
f=right(e)
f=left(v)
v
right(v)=g
g=left(e)
low(e)
low(f)

Figure 2. Properties of st-graphs.

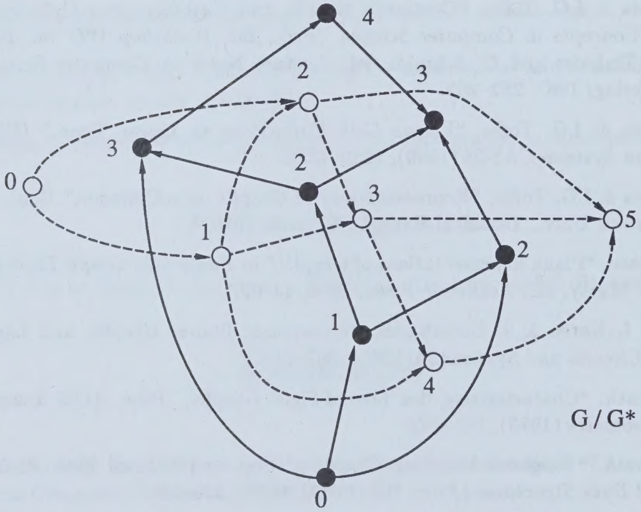

4
2
3
3
0
2
3
1
5
2
1
4
G / G*
0

Figure 3. A planar st-graph G (solid lines) and its dual G* (dashed lines).

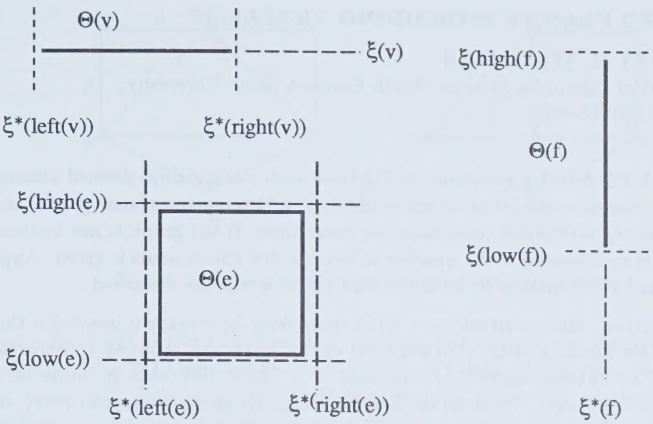

Θ(v)
ξ(v)
ξ*(left(v))
ξ*(right(v))
ξ(high(f))
Θ(f)
ξ(high(e))
Θ(e)
ξ(low(f))
ξ(low(e))
ξ*(left(e))
ξ*(right(e))
ξ*(f)

Figure 4. Computing the coordinates of the tiles.

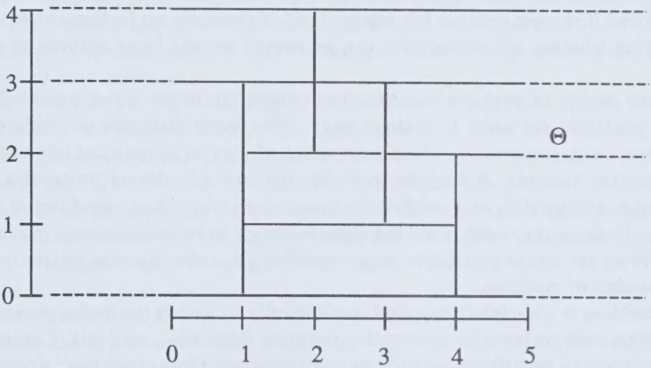

4
3
2
1
0
Θ
0
1
2
3
4
5

Figure 5. The tessellation representation on the plane computed by our algorithm for the planar st-graph G of Figure 3.

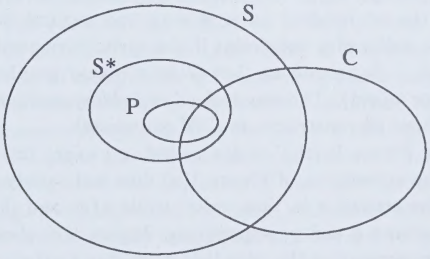


Figure 6. Relations between classes of visibility graphs. P, C, and S denote the classes of graphs that admit a visibility representation on the plane, cylinder, and sphere, respectively. S* denotes the class of graphs that admit an open visibility representation on the sphere.

CONSTRAINED PLANAR EMBEDDING PROBLEMS [1]

MATTHIAS F.M. STALLMANN
Department of Computer Science, North Carolina State University,
Raleigh, NC 27695-8206

Abstract. A *PQ-forest*, a collection of PQ-trees with relationships defined among interior nodes, is used to represent the set of all embeddings of a biconnected planar graph, and to find embeddings satisfying additional constraints in linear time. If the graph is not biconnected an algorithm which is exponential in the number of biconnected components is given. Applications to VLSI layout and relationships to book embedding problems are discussed.

1. Introduction. Recent advances in VLSI technology have made it inevitable that circuit layout be organized hierarchically. At each level of the hierarchy, modules constructed at the next lower level are interconnected. Sometimes it is highly desirable to route all of these connections on a single layer. The usual method for doing this assumes that the *pins* (points that are to be connected to other modules) around each module are in some prespecified sequence, determined by the design at the next lower level. For example, Marek-Sadowska and Tarng [12] give a model for single layer routing within a region containing modules, assuming that the pin order around each module is given in advance. The likelihood of routing all connections on one layer is improved if the pin order is left unspecified. If there are no further constraints, the problem of checking whether all connections can be routed on one layer reduces to planarity testing.

To simplify the design of modules in a situation where pin order is not prespecified, it is advantageous to constrain pin order in various ways. This paper discusses two different types of constraints: *contiguity constraints*, which force a set of pins to appear contiguously on the module boundary (for example, it may be desirable to have all address ports of a memory module appearing contiguously), and *uniformity constraints*, which force modules of the same type to adhere to the same pin order (allowing a single design to be used for each module type). Constraints like these are not as restrictive as prespecified pin order, but are restrictive enough to facilitate the design of modules.

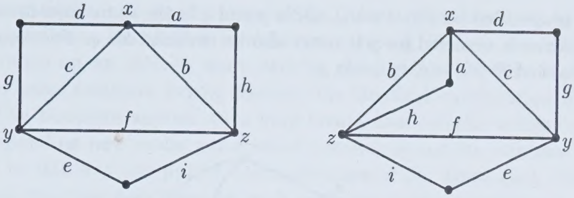
A planar embedding is completely specified topologically by giving the cyclic permutation of the edges incident on each vertex (this observation is due to Edmonds – see [16]). Contiguity and uniformity constraints can be defined formally as constraints on these cyclic edge permutations. A contiguity constraint is a subset $\{e_1, \ldots, e_k\}$ of the edges incident on a vertex v, and is satisfied if $e_1, \ldots, e_k$ appear contiguously around v in the embedding. A uniformity constraint, is a pair of edges lists $(v : e_1 \ldots e_d, w : f_1 \ldots f_d)$ with $\{e_1, \ldots, e_d\}$ being the set of edges incident on vertex v and $\{f_1, \ldots, f_d\}$ the set incident on $w \neq v$ (v and w must have the same degree). An embedding satisfies the uniformity constraint if the cyclic permutation of $e_1 \ldots e_d$ around v is the same as that of $f_1 \ldots f_d$ around w. Let G be a planar graph and let C be a set of constraints of either type (or mixed). The *constrained embedding problem* for G and C is to find an embedding of G that obeys all constraints in C (if one exists).

For the graph shown in Figure 1, let $C = \{(x : abcd,\ y : efgc), (x : abcd,\ z : ibfh)\}$, two uniformity constraints. The embedding of Figure 1(a) does not satisfy either constraint: the edges $abcd$ appear clockwise around x in that order, while $efgc$ and $ibfh$ must be permuted to obtain the embedding around y and z, respectively. Figure 1(b) shows an embedding that satisfies both constraints by permuting the edge lists according to the permutation 1243 – the clockwise sequences are $abdc$, $efcg$, and $ibhf$, respectively.

The results of this paper are that the constrained embedding problem can be solved in (a) linear time if G is biconnected, or (b) polynomial time if the number of biconnected components is bounded by a constant. In the time bounds cited later, m is the number of edges in

---

[1] This research is partially supported by the Office of Naval Research under contract N00014-88-K-0555



(a) (b)

Figure 1: Example of a constrained planar embedding

G and $m_{\mathcal{C}}$ is the total number of edges in $\mathcal{C}$. Section 2 introduces the concept of a PQ-forest, a generalization of a PQ-tree used in the algorithms. Section 3 describes the algorithm used to obtain result (a). Section 4 gives the ideas used for result (b). Section 5 discusses the relationship between constrained planar embedding and an NP-hard book embedding problem.

2. PQ-Forests. A *PQ-tree* (see [2]) specifies a set of admissible permutations on a ground set S, in this case the set of edges incident to a particular vertex. The leaves of a PQ-tree represent elements of S. In addition, there are two types of interior nodes. A *P-node* specifies that its children may appear in any order, but all leaf descendants of the P-node must appear consecutively. A *Q-node* specifies that its children must appear in the order they are listed or the reverse of that order.

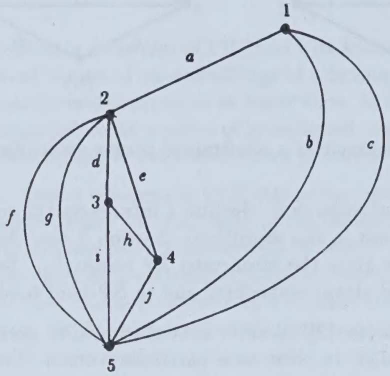
In our notation for PQ-trees, angle brackets <> enclose the children of a P-node, while square brackets are used to enclose the children of a Q-node. For example, $[<a\ b\ c>\ d\ e]$ denotes a PQ-tree whose root is a Q-node with three children: a P-node and the leaves d and e. The P-node has three children: the leaves a, b, and c. Among the admissible permutations of the five leaves are *abcde*, *edabc* (reverse children of Q-node), and *bacde* (permute children of the P-node). Not admissible are the permutations *dabce* (violates constraint imposed by Q-node) and *abdce* (violates P-node constraint). We use the notation $L(T)$, where T is a PQ-tree, to denote the set of all permutations of the leaves of T consistent with T (using the above rules), and $leaves(p)$ to denote the set of leaves in the subtree rooted at p.

We make use of two operations defined on PQ-trees. $Reduce(T, A)$ modifies T so that the elements of A are contiguous in every $\lambda \in L(T)$. $Intersect(T, T')$ modifies T and T' so that every $\lambda \in L(T)$ is also in $L(T')$, assuming T and T' are defined over the same ground set. The time bound for *Reduce* is $O(|A|)$. *Intersect*, introduced in [8], has a straightforward linear-time (in the number of leaves) sequential implementation. If the constraints cannot be satisfied then the results of *Reduce* and *Intersect* are defined to be empty trees.

PQ-trees can also be extended to allow cyclic shifts of the leaves in selected subtrees. If p is a *cyclic node* (in addition to being either a P-node or Q-node), any cyclic shift of $leaves(p)$ is allowed. Minor modifications allow us to perform the operations *Reduce* and *Intersect* (same time bounds) in the presence of cyclic nodes. See [8] for a strategy based on tree rotations.

The algorithms in this paper all work essentially as follows. First compute, for each vertex i, the PQ-tree T_i representing clockwise sequences of the edges around i that are consistent with some planar embedding. Such sequences are called *embedding consistent* sequences. Then use *Reduce* to apply additional contiguity constraints or *Intersect* to apply uniformity constraints.

Unfortunately the PQ-trees for distinct vertices are not independent of each other. The interactions between them must be modeled in some way. A *PQ-forest* $F = (\mathcal{T}, \mathcal{P})$ consists of a collection of PQ-trees $\mathcal{T} = (T_1, \ldots, T_n)$ and a partition $\mathcal{P}$ of the interior nodes of $T_1 \cup \ldots \cup T_n$. Two P-nodes in the same class of $\mathcal{P}$ have the same number of children and are constrained by the fact that corresponding children must always appear in the same sequence (the correspondence between children is specified by F). Two Q-nodes p and q in the same class must "flip" together, if the order of children is reversed for p it must also be reversed for q. For convenience $class(p)$ will denote the class of $\mathcal{P}$ containing node p.



$$\begin{aligned}
T_1 &= < a\ b\ c >_1 \\
T_2 &= << f\ g\ [d\ e]_2 >_3 a >_4 \\
T_3 &= < [i\ h]_5\ d >_6 \\
T_4 &= < j\ [h\ e]_7 >_8 \\
T_5 &= << f\ g\ [i\ j]_9 >_{10}\ b\ c >_{11}
\end{aligned}$$

Nodes 1,4,6,8, and 11 are cyclic. P-node classes are {1,11} and {3,10}. The only Q-node class is {5,7,9}.

Figure 2: A planar graph and its PQ-forest

Figure 2 shows a planar graph G and F_G, a PQ-forest representing the set of all possible embeddings of G. The root of every tree in F_G is a cyclic node. The equivalence classes can be obtained by a systematic reduction of G to a single edge using two transformations: (a) replace a triconnected component by a single edge (corresponds to a class of Q-nodes) and (b) replace a *branch graph*, graph consisting of $h \geq 2$ disjoint paths, by a single edge (corresponds to a class of P-nodes with h children). The theoretical basis for such a reduction was introduced by MacLane [10], who used it to count distinct embeddings. A straightforward modification of the PQ-tree planarity testing algorithm [2] can be used to compute F_G in linear time (see [14] for details; the structure of F_G depends on the *st*-numbering used, but the set of embeddings depends only on G). The planarity testing algorithm models constraints on the left-to-right sequence of edges to lower or higher numbered vertices. Reversing the order of the edges to lower numbered vertices gives a clockwise ordering around each vertex (the edges that need to be reversed are always contiguous — see [9]).

F_G can be used to compute the number of distinct embeddings of G or to enumerate all embeddings. Results on these problems (without using F_G) are reported in [3, 4]. Our first result uses F_G to compute a planar embedding subject to contiguity constraints.

THEOREM 2.1. *If G is biconnected planar graph and $\mathcal{C}$ consists only of contiguity constraints, constrained embedding for G and $\mathcal{C}$ can be solved in linear time ($O(m + m_{\mathcal{C}})$).*

Proof Sketch. Compute F_G in linear time and then do a series of *Reduce* operations corresponding to constraints in C on the trees of F_G. Constraints imposed by the classes of $\mathcal{P}$ are maintained as follows. Recall that modifications during *Reduce* are defined by *templates*, local transformations on an interior node and its children (see [2] for details). If p is a P-node modified by some template during *Reduce*, the identical modification is applied to every $q \in class(p)$. The template applied to p may create new interior nodes between p and its children. Corresponding new nodes are created between q and its children. Two new nodes that correspond in this way are put in the same class of $\mathcal{P}$. Since each P-node class in F_G has no more than two members (one for each endpoint of the corresponding branch graph), these modifications do not violate the time bound. Q-node classes are maintained off-line using a graph G_Q. Each vertex of G_Q represents a Q-node and an edge $\{p, q\}$ is added whenever the *Reduce* algorithm detects that p and q should be in the same class (for example, when a template merges the children of p with the children of q). The edge has cost 0 if p and q have the same orientation, 1 if p must flip in order to be consistent with q. The connected components of G_Q correspond to Q-node classes in $\mathcal{P}$ and the constraints are satisfiable as long as G_Q has no cycles of odd cost. In the sequential implementations of *Reduce* [2, 4], it may not be possible to determine the relative orientation of two Q-nodes whose children are merged (without violating the linear time bound). Thus only a subgraph of G_Q is maintained during the *Reduce* operations. Q-nodes whose relative orientation was unable to be determined leave behind enough information so that a final pass through all trees can reconstruct the remainder of G_Q (the algorithm in [4] maintains information about relative orientation using "direction indicators"; a simpler strategy using only parent and sibling pointers is given in [14]; the latter allows constraints to be processed on-line, with an additional α factor in the time bound). □

3. Arbitrary Constraints on Biconnected Graphs. Additional complications arise when uniformity constraints are included. A uniformity constraint $(v : e_1 \ldots e_d,\ w : f_1 \ldots f_d)$ is enforced by intersecting T_v and T_w in F_G (using $1, \ldots, d$ as the ground set for both T_v and T_w). The modifications to T_v and T_w may affect other trees in the forest. In particular, the children of a P-node p in T_v may be forced to obey the same constraints as the children of a P-node q in T_w. If p and q were originally in distinct equivalence classes of F_G, these classes need to be merged. Thus it is no longer the case that each class of P-nodes has cardinality 2.

The basic *Intersect* algorithm, a sequential implementation of that in [8], is shown in Figure 3. The operation *Replace*(T, p, q) replaces the subtree rooted at p in T by a new subtree rooted at q. A *pseudonode* is the (virtual) parent of a group of contiguous children of the same Q-node and behaves like a Q-node (see [2]). Figure 4 shows the routine *Propagate_Constraints*, which ensures that constraints imposed by classes of $\mathcal{P}$ are propagated correctly (assuming that this is true for *Reduce*, as described in the proof of Theorem 2.1). *Intersect* in the context of a PQ-forest not only forces the structures of T and T' to conform, but also causes creates new class relationships among the interior nodes of T and T'.

There are three ways in which a uniformity constraint can fail to be satisfied. The first of these is a *contiguity conflict*, detected during one of the *Reduce* operations. For example, in Figure 5 the uniformity constraint fails when *Intersect*(T_x, T_y) leads to *Reduce*$(T_x, \{2, 4\})$, which fails.

A *P-node conflict* arises whenever corresponding children of two P-nodes in the same class cannot be sequenced without violating the uniformity constraint (detected at (1) of the routine *Propagate_Constraints*, see Figure 4). The simplest case of this is illustrated in Figure 6; the cyclic P-node roots of T_x and T_y belong to the same class in F_G.

A *Q-node conflict* occurs when the relative orientation of two Q-nodes is inconsistent (detected by finding an odd cycle in G_Q). In Figure 7, Q-nodes p and q are constrained to have the same orientation by F_G, but are forced to have the opposite orientation by the uniformity constraint. Two edges between p and q are added to G_Q, one with label 1 during the construction of F_G and another with label 0 during *Intersect*(T_x, T_y) (at (2) in *Propagate_Constraints*).

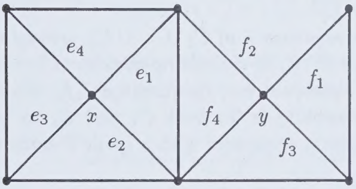
The time bound of the algorithm in Figures 3 and 4 appears to be $O(dk)$, where d is

```
procedure $Intersect(T, T')$ modifies $T$
    /* Makes $L(T) = L(T) \cap L(T')$; $T'$ is not modified
       (modification of $T'$ can easily be accomplished by
        substituting $T$ for it at the end) */
    if $T$ has just one node then return endif
    $p :=$ any node of $T'$ whose children are leaves
    /* Note: in what follows $leaves(p)$ is shorthand for the
        set of elements represented by $leaves(p)$, i.e. the
        labels on $leaves(p)$ */
    $A := leaves(p)$
    $Reduce(T, A)$
    if $p$ is a Q-node and $leaves(p)$ are in the sequence $a_1, a_2, \ldots, a_k$ then
        $Reduce(T, \{a_1, a_2\})$
        $Reduce(T, \{a_2, a_3\})$
            $\ldots$
        $Reduce(T, \{a_{k-1}, a_k\})$
    endif
    $q :=$ the node (or pseudonode) of $T$ with $leaves(q) = A$
    $Propagate\_Constraints(p, q)$
    /* allow cyclic shifts for $leaves(q)$ only if they were also
         allowed for $leaves(p)$ */
    $cyclic(q) := cyclic(p)$ and $cyclic(q)$
    /* constraints for $T_p$, the subtree rooted at $p$ in $T'$ have now
         been incorporated into $T_q$, the subtree rooted at $q$ in $T$;
         the remainder of $T$ is now made to conform with the
         remainder of $T'$ */
    save $T_q$
    $l, l'$ are new leaves with identical labels
    $Replace(T, q, l)$; $Replace(T', p, l')$
    $Intersect(T, T')$
    /* restore $T_q$ by splicing it back onto $T$ */
    $Replace(T, l, q)$
end $Intersect$
```

Figure 3: Sequential *Intersect* algorithm for PQ-forests

```
procedure Propagate_Constraints($p, q$)
    if $p$ is a P-node then
        if $q$ is a P-node and $q$ has leaf children then
            if $q \in class(p)$ then
(1)             check to see if labels on corresponding children agree
                    (up to cyclic shifts if $p$ and $q$ are both cyclic)
            else merge $class(p)$ with $class(q)$ endif
        else /* $T_q$ has a more complicated structure than $T_p$ */
            /* Make all children of nodes in $class(p)$ subject to the
               constraints of $T_q$ */
            for $r \in class(p)$ do
                $s :=$ parent($r$); $T(r) :=$ the tree in which $r$ appears
                create a copy of $T_q$ whose leaves are the children of $r$
                    (put new nodes in the copy into the same equivalence class
                    as corresponding nodes of $T_q$)
                $q' :=$ root of the copy
                Replace($T(r), r, q'$)
            end do
        endif
    else /* $p$ is a Q-node */
(2)     merge $class(p)$ with $class(q)$ (keep track of relative orientation)
    endif
end Propagate_Constraints
```

Figure 4: Routine for updating class relationships during *Intersect*



$T_x = < e_1\ [e_2\ e_3\ e_4] >$ $\qquad$ $T_y = <<< f_4\ f_2 > f_3 > f_1 >$

uniformity constraint is $(x : e_1 e_2 e_3 e_4,\ y : f_1 f_2 f_3 f_4)$

Figure 5: A contiguity conflict

the number of leaves in T and k is the maximum number of nodes in any class of P-nodes. However, any access to a P-node p in the inner loop of *Propagate_Constraints* is accompanied by a structural change, either the insertion of new interior nodes between p and its children, or the replacement of p by a Q-node, or both. Since Q-nodes are never replaced by P-nodes and the total number of interior nodes in F_G is always linear in the number of leaves (an invariant of PQ-trees – see [2]), a linear time bound results (as in Theorem 2.1 constraints may be processed on-line with an α penalty in the time bound).

THEOREM 3.1. *If G is biconnected and $\mathcal{C}$ contains both contiguity and uniformity constraints, constrained embedding for G and $\mathcal{C}$ can be solved in linear time.* □

In the VLSI applications it is desirable to allow modules to "flip", i.e. two different instantiations of the same module are allowed to have reversed pin order wrt each other. A modified problem definition allows a *reversible uniformity constraint* $(v : e_1 \dots e_d,\ w : f_1 \dots f_d)_R$ to be met as long as the cyclic permutation of $f_1 \dots f_d$ is either identical to or the reverse of that for $e_1 \dots e_d$. Constrained embedding with reversible uniformity constraints appears to be more difficult than one in which no reversal is allowed. The issue of reversal only comes up when handling P-node and Q-node conflicts (allowing reversal does not help if failure to meet the constraint is the result of a contiguity conflict). The nontrivial part of handling reversible uniformity constraints occurs during the processing of G_Q to determine whether Q-node conflicts exist. Labels on edges of G_Q that were added (in line (2) of *Propagate_Constraints*) because of a reversible uniformity constraints can be complemented as long as this is done simultaneously for all edges corresponding to the same constraint. What results is a graph problem that may be formulated as follows. Given a graph $G_Q = (V_Q, E_Q)$ with edges labeled 0 or 1, and a collection $\mathcal{C}_Q = \{E_1, \dots, E_k\}$ of subsets of E_Q, find a subcollection $\mathcal{C}'$ of $\mathcal{C}_Q$ so that if the label on every edge appearing in $\mathcal{C}'$ is complemented, the resulting graph has no cycles of odd length. This can be solved in polynomial time ($O(m_{\mathcal{C}}{}^{2.37})$) by solving a system of equations over $GF(2)$ (each equation represents a fundamental cycle in G_Q).

4. Multiple Biconnected Components. When multiple biconnected components exist our approach is to compute PQ-trees for embedding-consistent edge sequences within each biconnected component and try out all different ways of combining these into a single PQ-tree around each vertex. Constraints are applied to each possible combination until one combination results in a successful embedding.

Suppose $(C_1, \dots, C_h)$ is the partition of edges incident to v into biconnected components (v is an articulation point if $h > 1$).

LEMMA 4.1. *A clockwise sequence λ of $C_1 \cup \dots \cup C_h$ around v is embedding-consistent iff (a) for each i, the subsequence $\lambda \cap C_i$ is embedding-consistent for the component containing C_i, and (b) λ does not contain a subsequence of the form $e_i e_j f_i f_j$, where $e_i, f_i \in C_i$ and $e_j, f_j \in C_j$.*

When $h = 2$ the only possibility is for both C_1 and C_2 to be contiguous in λ. We can represent all embedding consistent sequences with a cyclic P-node whose children are the roots of the trees for C_1 and C_2.

THEOREM 4.2. *If at most two biconnected components are joined at any articulation point of G, then constrained embedding for G and $\mathcal{C}$ can be solved in linear time.*

When $h > 2$, the following procedure defines a PQ-tree for embedding consistent edge sequences around v: (0) choose two components C_1 and C_2 and construct a PQ-tree for them as in the previous discussion (let $C^* = C_1 \cup C_2$), (1) choose a new component $C_i \not\subseteq C^*$, (2) choose an edge $e \in C^*$ (let C_j be the component containing e), (3) constrain the edges of C_i to occur between e and a neighbor of e in an embedding-consistent sequence for C_j, and (4) repeat steps (1)-(3) until all components are chosen. To construct a PQ-tree, suppose $\mathcal{S}(e)$ is the set of components chosen for e in step (3) and $T(e)$ is the PQ-tree for the component containing e. Let $\{T_1, \dots, T_k\}$ be the set of PQ-trees for components in $\mathcal{S}(e)$. Then the constraints of (3) are modeled by doing *Replace*$(T(e), e, < e\ T_1 \dots T_k >)$. It is not hard to show that the above procedure always defines a single PQ-tree T and that a sequence λ of edges around v is embedding-consistent iff $\lambda \in L(T)$ (this procedure cannot be used to count embeddings because some λ's are in more than one $L(T)$ – see [3] for a counting method).

The number of ways to construct the PQ-tree when v has degree d is $O(d^{h-2})$. If we do this for each vertex independently, we get the bound in the following theorem. If $h(v)$ is the number of biconnected components meeting at vertex v in $G = (V, E)$, let $\beta(G) = \sum_{v \in V} \max(0, h(v) - 2)$. Note that $\beta(G)$ is linear in the number of biconnected components of G (because the bipartite graph relating articulation points to biconnected components is a tree and $h(v)$ is the number of edges incident on v in this graph).

THEOREM 4.3. *Constrained embedding for G can be solved in $O(d^{\beta(G)}(m + m_{\mathcal{C}}))$ time, where d is the maximum degree in G.* □

5. Relationship to Book Embedding. The *book embedding problem* for a graph G is the problem of embedding the vertices of G on the spine of a book and assigning the edges of G to pages in such a way that edges mapped to the same page do not cross. Book embedding was introduced by Bernhart and Kainen [1]. Applications include VLSI design [13, 5] and complexity theory [6]. Previous work has concentrated on bounding the *pagenumber* of G for various restricted graph classes, where the pagenumber is the smallest number of pages required for a book embedding of G [18, 7, 11]. The following problem, the decision version of book embedding, is known to be NP-complete even if G is planar and the pagenumber is 2 [17]: Given a graph $G = (V, E)$ (assume $V = \{1, \ldots, n\}$) and a pagenumber k, does there exist a partition of $E = (E_1, \ldots, E_k)$ and a (cyclic) permutation π of V, so that $G'_j = (V, E_j \cup C)$, where $C = \{\{\pi(i), \pi((i \bmod n) + 1)\} \mid 1 \leq i \leq n\}$, is outerplanar for $1 \leq j \leq k$?

A non-deterministic algorithm for book embedding must make two choices: (a) a partition of E into pages, and (b) a permutation π of V along the spine. Fixing π in advance does not appear to make book embedding significantly easier: book embedding with fixed π is equivalent to circle graph k coloring which is NP-complete for any $k \geq 4$ [15]. The case $k = 3$ is still open and $k = 2$ is trivial. Not much is known about *fixed partition book embedding (FPBE)*, book embedding with the edge partition fixed.

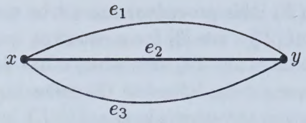
LEMMA 5.1. *FPBE reduces to constrained planar embedding in linear time (or log space).*

Proof Sketch. Given a k-layer FPBE problem on $G = (V, E)$, we can define a planar graph $G'' = (V'', E'')$ and a set of uniformity constraints $\mathcal{C}$ relating vertices in V'' so that G has a book embedding wrt $(E_1, \ldots, E_k)$ iff G'' has a constrained embedding wrt $\mathcal{C}$. For $j = 1, \ldots, k$ let $V''_j = \{v_j\} \cup \{w_{i,j} \mid i \in V\}$ and $E''_j = \{e_{ij} = \{v_j, w_{i,j}\} \mid i \in V\} \cup \{\{w_{i_1,j}, w_{i_2,j}\} \mid \{i_1, i_2\} \in E_j\}$. Now let G'' be the disjoint union of components $G''_j = (V''_j, E''_j)$. Let

$$\begin{aligned} \mathcal{C} \;=\; & \{(e_{11}e_{21}\ldots e_{n1}, e_{12}e_{22}\ldots e_{n2}), \\ & (e_{11}e_{21}\ldots e_{n1}, e_{13}e_{23}\ldots e_{n3}), \\ & \ldots, \\ & (e_{11}e_{21}\ldots e_{n1}, e_{1k}e_{2k}\ldots e_{nk})\}. \end{aligned}$$

Figure 8(a) shows G and a partition of E into two layers for a typical FPBE problem. Figure 8(b) shows the resulting G''. □

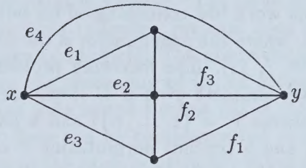
Lemma 5.1 has two consequences. First, it shows that constrained embedding is at least as hard as a variant of book embedding, which, though not known to be NP-hard, appears to be difficult. Second, some restrictions on fixed partition book embedding are known to make the problem polynomially solvable. For example, Theorem 3.1 implies that fixed partition book embedding can be solved in linear time if each partition is a connected graph (note that every connected component of E_j becomes a biconnected component of G''_j). Similarly, FPBE can be solved in polynomial time if the number of connected components in each E_j is bounded by a constant. While these observations are not of any practical significance for book embedding, they do rule out many strategies for proving FPBE NP-hard.



$T_x =< e_1\ e_2\ e_3 >$ $T_y =< e_1\ e_2\ e_3 >$

uniformity constraint is $(x : e_1e_2e_3,\ y : e_1e_2e_3)$

Figure 6: A P-node conflict


e_4
e_1
f_3
x
e_2
y
f_2
e_3
f_1

$T_x =< [e_1\ e_2\ e_3]_p\ e_4 >$ $T_y =< [f_1\ f_2\ f_3]_q\ e_4 >$

uniformity constraint is $(x : e_1e_2e_3e_4,\ y : f_1f_2f_3e_4)$

Figure 7: A Q-node conflict

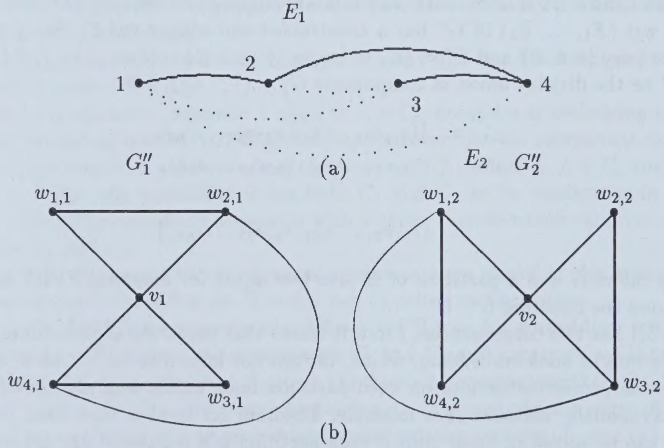

E_1
2
1
4
3
G_1''
(a)
E_2
G_2''
$w_{1,1}$
$w_{2,1}$
$w_{1,2}$
$w_{2,2}$
v_1
v_2
$w_{4,1}$
$w_{3,1}$
$w_{4,2}$
$w_{3,2}$
(b)

Figure 8: Reduction of FPBE to UE

REFERENCES

[1] F. Bernhart and B. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory B*, 27:320 – 331, 1979.

[2] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *Journal of Computer and System Sciences*, 13:335 – 379, 1976.

[3] J. Cai, X. Han, and R.E. Tarjan. New solutions to four planar graph problems. Typescript.

[4] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using *PQ*-trees. *Journal of Computer and System Sciences*, 30:54 – 76, 1985.

[5] F. Chung, T. Leighton, and A. Rosenberg. Embedding graphs in books: A layout problem with applications to VSLI design. *SIAM Journal on Algebraic and Discrete Methods*, 8:33 – 58, 1987.

[6] Z. Galil, R. Kannan, and E. Szemeredi. On non-trivial separators for k-page graphs and simulations by nondeterministic one-tape Turing machines. In *Proceedings 18th Annual Symposium on Theory of Computing*, pages 39 – 49, 1986.

[7] L.S. Heath. Embedding outerplanar graphs in small books. *SIAM Journal on Algebraic and Discrete Methods*, 8:198 – 218, 1987.

[8] P.N. Klein and J.H. Reif. An efficient parallel algorithm for planarity. *Journal of Computer and System Sciences*, 37:190 – 246, 1989.

[9] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In P. Rosenstiehl, editor, *Theory of Graphs: International Symposium, July, 1966*, pages 215 – 232, New York, 1967. Gordon and Breach.

[10] Saunders MacLane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, pages 460 – 472, 1937.

[11] S.M. Malitz. Genus g graphs have pagenumber $O(\sqrt{g})$. In *Proceedings 29th Annual Symposium on Foundations of Computer Science*, pages 458 – 468, 1988.

[12] M. Marek-Sadowska and T.T-K. Tarng. Single-layer routing for VLSI: Analysis and algorithms. *IEEE Transactions on Computer Aided Design*, CAD-2:246 – 259, 1983.

[13] A. Rosenberg. The Diogenes approach to fault-tolerant arrays of processors. *IEEE Transactions on Computers*, C-32:902 – 910, 1983.

[14] M. Stallmann. Using *PQ*-trees for planar embedding problems. Technical Report TR-85-24, NCSU-CSE, 1985.

[15] W. Unger. On the k-colouring of circle graphs. In *Proceedings 5th Annual Symposium on Theoretical Aspects of Computer Science*, pages 61 – 72, 1988.

[16] Arthur T. White. *Graphs, Groups, and Surfaces.* North-Holland, 1984.

[17] A. Wigderson. The complexity of the Hamiltonian circuit problem for maximal planar graphs. Technical Report 298, EECS Department, Princeton University, 1982.

[18] M. Yannakakis. Four pages are necessary and sufficient for planar graphs. In *Proceedings 18th Annual Symposium on Theory of Computing*, pages 104 – 108, 1986.

AN EFFICIENT ALGORITHM FOR DOMINATION IN PERMUTATION GRAPHS

C. RHEE, S. K. DHALL and S. LAKSHMIVARAHAN
School of Electrical Engineering and Computer Science
University of Oklahoma, Norman, OK 73019
H. KIM
Department of Mathematics and Computer Science
University of Tennessee, Martin, TN 38238

ABSTRACT

In this paper, we present an $O(n \log n)$ algorithm for finding a minimum weight independent dominating set in a weighted permutation graph. The key idea is to reduce this problem to the shortest path problem in a weighted digraph.

1. INTRODUCTION

Let $<n> = \{1, 2, ..., n\}$ and $\Pi = (\Pi_1 \Pi_2 \cdots \Pi_n)$ be a permutation on $<n>$. Then the *permutation* graph of Π, denoted by $G(\Pi) = G(V, E)$, is defined as follows: Let $V = <n>$ and $E = \{i\ j \mid (i - j)(\Pi^{-1}{}_i - \Pi^{-1}{}_j) < 0\}$, where $\Pi^{-1}{}_i$ is the position of vertex i in the sequence $(\Pi_1 \Pi_2 \cdots \Pi_n)$. In a more pictorial way of representing $G(\Pi)$, we write the vertices 1, 2, ..., n horizontally from left to right and, underneath them, write the numbers $\Pi_1, \Pi_2, ..., \Pi_n$ in sequence, again horizontally from left to right. In this matching diagram the line connecting two i's intersects the line connecting two j's if and only if $i\ j \in E$ [5, 6, 7]. A subset D of V is called a *dominating* set for G if every vertex not in D is adjacent to at least one vertex in D. A dominating set D is said to be *independent* if no two vertices in D adjacent to each other. A graph G is called a *weighted* graph if there exists a weight function $w : V \rightarrow R$ (the set of real numbers), where $w(u)$ is called the weight of node u. An independent dominating set $D^* \subseteq V$ such that $S(D^*) = \min_{D \subseteq V}\{S(D)\}$, where D is an independent dominating set and $S(D) = \sum_{u \in D} w(u)$, is called a *minimum weight* independent dominating set in G.

Since the permutation graph was introduced by Even, Pnueli and Lempel for the first time [5, 8], various interesting problems for this graph have been investigated by many authors [1, 2, 3, 5, 6, 8, 9, 10]. Pnueli, Lempel and Even [8] describe an $O(n^3)$ algorithm for testing if a given undirected graph G is a permutation graph. Spinrad [9] improved the above result by deriving an $O(n^2)$ algorithm for orienting comparability graphs. A proper subset of the permutation graphs called cographs has been studied in Corniel, Perl and Stewart [4]. Bipartite permutation graphs are analyzed by Spinrad, Brandstadt and Stewart [10]. In particular, for the domination problem, Farber and Keil [6] introduced an $O(n^2)$ time algorithm for the minimum cardinality dominating set problem in permutation graphs. They also gave $O(n^3)$ time algorithms for finding a minimum weight dominating set and a minimum weight independent dominating set. Later, Brandstadt and Kratsch [3] improved one of the results in [6] by introducing an $O(n^2)$ time algorithm for the latter problem. The same problem was also discussed by Atallah, Manachar and Urrutia [2] who discovered a more efficient $O(n \log^2 n)$ time algorithm. Recently, Atallah and Kosaraju [1] present an $O(n \log n)$ time algorithm for the *maxdominance* problem to which the minimum weight independent dominating set problem for permutation graphs can be reduced in linear time.

In this paper we present an $O(n \log n)$ time algorithm for finding a minimum weight independent dominating set in permutation graphs in a different way. Our approach consists in reducing this problem to the well known shortest path problem for a related digraph.

2. MINIMUM WEIGHT INDEPENDENT DOMINATING SET

Let $G = (V, E)$ with $|V| = n$ be a weighted permutation graph. For each $i \in V$, we define relations $NextL(i)$ and $NextR(i)$ in the following way: Let $NextR(i) = \{ j \in V \mid i < j,\ \Pi^{-1}_i < \Pi^{-1}_j,$ and there is no vertex $k \in V$ such that $i < k < j$ and $\Pi^{-1}_i < \Pi^{-1}_k < \Pi^{-1}_j \}$ and $NextL(i) = \{ j \in V \mid i > j,\ \Pi^{-1}_i > \Pi^{-1}_j,$ and there is no vertex $k \in V$ such that $i > k > j$ and $\Pi^{-1}_i > \Pi^{-1}_k > \Pi^{-1}_j \}$. We also introduce sets $Leftclique(\Pi) = \{ i \in V \mid$ there is no vertex $j \in V$ such that $i > j$ and $\Pi^{-1}_i > \Pi^{-1}_j \}$ and $Rightclique(\Pi) = \{ i \in V \mid$ there is no vertex $j \in V$ such that $i < j$ and $\Pi^{-1}_i < \Pi^{-1}_j \}$. We construct a weighted digraph $H = (V^H, E^H)$ from $G = (V, E)$ in the following way: Let $V^H = V$, and for each $i, j \in V^H$, $(i, j) \in E^H$ iff $i \in NextL(j)$. Then we have the following lemma.

LEMMA 1: Any independent dominating set for G corresponds to a directed path from a member of $Leftclique(\Pi)$ to a member of $Rightclique(\Pi)$ in H. Throughout the paper, we use the notation *LRpath* to indicate any of such paths.

PROOF: Let $p = i_1 i_2 \cdots i_t$ be a *LRpath* in H and $j \in V$ be a vertex which does not appear in p. Suppose $j\, i_s \notin E$ for each $s = 1, 2, \ldots, t$. Then there must be a vertex i_k in p, $1 \le k \le t-1$, such that $i_k < j < i_{k+1}$, $\Pi^{-1}_{i_k} < \Pi^{-1}_j < \Pi^{-1}_{i_{k+1}}$, and $(i_k, i_{k+1}) \in E^H$. However, this is not allowed in the construction of E^H since $i_k \notin NextL(i_{k+1})$. Therefore the set $V(p) = \{i_1, i_2, \ldots, i_t\}$ is a dominating set for G and $V(p)$ is clearly independent from the relation *NextL*. For the other direction, suppose D be an independent dominating set for G. Let R be a relation on D such that $i\ R\ j$ iff $i < j$ and $\Pi^{-1}_i < \Pi^{-1}_j$. Then for each $i, j \in D$, either $i\ R\ j$ or $j\ R\ i$ since D is an independent set. That is, R is a total relation on D. Therefore, there is a linear ordering $< D >$ on D and this linear chain corresponds to a *LRpath* in H since every ordered pair of two adjacent vertices in $< D >$ satisfies the relation *NextL*.

For each $i \in V$, define $FirstsmallR(i)$ to be the first node which is smaller than i, going from i to the right side of i on Π, and $BiggestsmallL(i)$ to be the largest node which is smaller than i and lies on the left side of i in Π.

LEMMA 2: The functions $FirstsmallR(i)$ and $BiggestsmallL(i)$ for all $i \in V$ can be computed in $O(n \log n)$ time.

PROOF: The following simple procedure will compute $FirstsmallR(i)$ for each $i \in V$: Visit $\Pi = (\Pi_1 \Pi_2 \cdots \Pi_n)$ from Π_1 to the right until Π_n is visited. For each Π_i, put Π_i into an *AVL* tree using its value as a key. Then, for every Π_i's successor j, $FirstsmallR(j) = \Pi_i$. Once a node in the tree gets its *FirstsmallR* value, it will be popped up from the tree. The entire procedure will take $O(n \log n)$ time. $BiggestsmallL(i)$ can be similarily computed.

Now, we give a sketch of an $O(n \log n)$ time algorithm for finding a shortest *LRpath* in H. In fact, our algorithm does not construct all the edges in H in a static way, which would require at least $O(n^2)$ time. Instead, only a small subset of E^H will be computed in a dynamic manner which guarantees that this sparse graph still holds a shortest *LRpath* for H. Since the algorithm works recursively on a smaller subpermutation, we introduce, for convenience, a notation to indicate a half of entire permutation: Let $\Pi_{1:n} = \Pi$, $\Pi_{1:n/2} = (\Pi_1 \Pi_2 \cdots \Pi_{n/2})$ and $\Pi_{n/2+1:n} = (\Pi_{n/2+1} \Pi_{n/2+2} \cdots \Pi_n)$, and let $V(\Pi_{1:n/2})$ denote the vertex set of $\Pi_{1:n/2}$. In our algorithm, we assume that initially every node in $Leftclique(\Pi)$ is linked to itself and all the other nodes in H are linked to an imaginary node whose weight is ∞. This can be done in linear time. We also assume that the functions *FirstsmallR* (i) and *BiggestsmallL* (i) are computed for each $i \in V^H$. (See Lemma 2.) The algorithm consists of 5 stages:

Stage 1: For the given $\Pi_{1:n}$, find recursively a shortest path from each node in *Leftclique* $(\Pi_{1:n/2})$ to each node in $V(\Pi_{1:n/2})$

Stage 2: For each node i of $\Pi_{1:n/2}$, let i memorize a node j such that the length of *subpath* (j), denoted by $|subpath(j)|$, is the minimum of $\{|subpath(k)| \mid k \in Clique(i) \cap V(\Pi_{1:n/2})\}$, where *subpath* (j) is a shortest path from *Leftclique* $(\Pi_{1:n})$ to j and *Clique* (i) is defined recursively in the following way: Let $Clique(i) = \{i\}$ if i does not have *FirstsmallR* (i). Otherwise let $Clique(i) = \{i\} \cup Clique(FirstsmallR(i))$.

This can be done in $O(n)$ steps since, for each i, we only need to compare the length of *subpath* (i) and the length of *subpath* (k), where k is a node memorized by *FirstsmallR* (i) in the previous step in this stage. When there is no *FirstsmallR* (i) for some i, let $i = j$. Note that *subpath* (i) and it length are already computed in Stage 1.

Stage 3: For each node i of $\Pi_{n/2+1:n}$, if $NextL(i) \cap V(\Pi_{1:n/2})$ is not empty, then link i to a node j such that $|subpath(j)| = Min\{|subpath(k)| \mid k \in NextL(i) \cap V(\Pi_{1:n/2})\}$.

This can be done in $O(n)$ steps since, for each such i, we just need to refer *BiggestsmallL* (i) which is also a member of *NextL* (i) and link i to the node memorized by *BiggestsmallL* (i). Recall that *BiggestsmallL* (i) keeps a shortest path from *Leftclique* $(\Pi_{1:n})$ to a node in *NextL* (i) in Stage 2. If there is no such *BiggestsmallL* (i), do nothing. When this stage is over, any node i in $V(\Pi_{n/2+1:n})$ such that *NextL* (i) contains some nodes in $V(\Pi_{1:n/2})$ will be linked to the node k, where $k \in NextL(i) \cap V(\Pi_{1:n/2})$ and length of *subpath* (k) is minimum over all nodes in $NextL(i) \cap V(\Pi_{1:n/2})$.

Stage 4: Find recursively a shortest path from each node in *Leftclique* $(\Pi_{n/2+1:n})$ to each node in $V(\Pi_{n/2+1:n})$. Note that at the end of Stage 3 every node in *Leftclique* $(\Pi_{n/2+1:n})$ is already linked to a shortest path starting from a node in *Leftclique* $(\Pi_{1:n})$.

Stage 5: Pick up a node from *Rightclique* $(\Pi_{1:n})$ which is linked to a shortest *LRpath*.

LEMMA 3: Above algorithm computes a shortest *LRpath* in H correctly.

PROOF: Use induction on n. Assume that the algorithm compute a shortest *LRpath* in a digraph $H_{1:n}$ constructed from $\Pi_{1:n}$ to prove for the digraph $H_{1:n+1}$ constructed from $\Pi_{1:n+1}$. Consider Π_{n+1} in $\Pi_{1:n+1}$. If Π_{n+1} is an isolated node in $\Pi_{1:n+1}$ i.e. $\Pi_{n+1} = \Pi^{-1}{}_{n+1}$, then Π_{n+1} will be easily linked to a shortest *LRpath* in $H_{1:n}$ in Stage 3. On the other hand, if Π_{n+1} is adjacent to some node in $\Pi_{1:n+1}$ and if $NextL(\Pi_{n+1})$ is not empty, then, by Stage 3 again, Π_{n+1} is linked to a node j in $NextL(\Pi_{n+1})$ such that $|subpath(i)|$ is minimum. (If $NextL(\Pi_{n+1})$ is empty, Π_{n+1} itself is a path.) Therefore, when Π_{n+1} is not a member of any shortest *LRpath* in $H_{1:n+1}$, a shortest *LRpath* in $H_{1:n}$ which the algorithm can compute correctly by the assumption is also a shortest *LRpath* in $H_{1:n+1}$. If Π_{n+1} is the last member of a shortest *LRpath* in H_{n+1}, then Π_{n+1} can be linked to a shortest *LRpath* of the digraph constructed from $\Pi^{(n+1)}{}_{1:n+1}$, where $\Pi^{(n+1)}{}_{1:n+1}$ is the resulting permutation after deleting Π_{n+1} and all the nodes adjacent to Π_{n+1} from $\Pi_{1:n+1}$.

We are now ready to state the following main result.

THEOREM 1: The minimum weight independent dominating set problem in a weighted permutation graph can be solved in $O(n \log n)$ time.

Acknowledgement

This presentation has greatly benefited from our discussion with Professor G. K. Manachar who pointed out a flaw in the original version of our paper and the existence of references [1] and [3].

REFERENCES

[1] M. J. Atallah and S. R. Kosaraju, "An Efficient Algorithm for Maxdominance, with Applications", Dept. of Comp. Sci., Purdue Univ, 1987.

[2] M. J. Atallah, G. K. Manachar and J, Urrutia, "Finding a Minimum Independent Dominating Set in a Permutation Graph", Disc. Math., Vol 21, pp 117 - 183, 1988.

[3] A. Brandstadt and D. Kratsch, "On Domination Problems for Permutation and Other Graphs", Theoretical Comp. Sci., Vol 54, pp 181 - 198, 1987.

[4] D. G. Corneil, Y. Perl and L. K. Stewart, "A linear recognition algorithm for cographs", SIAM J. Comput., Vol 14, pp 926 - 934, 1985.

[5] S. Even, A. Pnueli and A. Lempel, "Permutation graphs and transitive graphs", J. ACM, Vol 19, pp 400 - 410, 1972.

[6] M. Farber and J. M. Keil, "Domination in permutation graphs", J. Algorithms, Vol 6, pp 309 - 321, 1985.

[7] M. Golumbic, *Algorithmic graph theory and perfect graphs,* Academic Press, New York, 1980.

[8] A. Pnueli, A. Lempel and S. Even, "Transitive orientation of graphs and identification of permutation graphs", Canad. J. Math., Vol 23, pp 160 - 175, 1971.

[9] J. Spinrad, "On comparability and permutation graphs", SIAM J. Comput., Vol 14, pp 658 - 670, 1985.

[10] J. Spinrad, A. Brandstadt and L. Stewart, "Bipartite permutation graphs", Discrete Applied Math., Vol 18, pp 279 - 292, 1987.

TR-MATCHING FOR CHORDAL GRAPHS

Jean R. S. Blair[†]
Department of Computer Science
University of Tennessee
Knoxville, TN 37996

S. S. Ravi [‡]
Department of Computer Science
SUNY at Albany
Albany, NY 12222

ABSTRACT

The problem of transmitter-receiver matching (TRM) in communication networks is addressed. It is shown that TRM is NP-complete in general for networks whose topologies correspond to chordal graphs. A linear time algorithm is given to solve TRM for topologies that belong to a subclass of chordal graphs.

1. INTRODUCTION

This extended abstract addresses an interesting problem that arises in conjunction with the diagnostic testing of communication networks. The topology of such a network can be modeled by an undirected graph $G(V,E)$ (without multiple edges or self loops) where V represents the set of sites (each with a transmitter and a receiver) and E represents the set of bidirectional links between the sites. During the maintenance period, each link of the network is tested by transmitting a signal from one end of the link and checking to see that the signal is properly received at the other end. Since the network is usually unavailable to the users during the maintenance period, it is important to complete the tests within a short period of time. One method of accomplishing this task is to carry out the tests over a small number of phases, with each phase simultaneously testing as many links as possible. In practice, the following constraints have to be taken into account while testing certain types of wireless communication networks [7, 6]. First, a site can act either as a transmitter or as a receiver but not both. Second, a transmitting site sends signals along only *one* of the links emanating from it. Finally, if signals from two or more transmitters reach a receiving site, they *interfere* with each other and the tests cannot be completed. Hence the set of links tested in each phase must be chosen in such a way that interference does not occur.

[†] Supported in part by the Mathematical Sciences Section of the Oak Ridge National Laboratory and a Faculty Research Award from the University of Tennessee.

[‡] Supported in part by NSF Grants DCI-8603318 and CCR-8905296.

In graph theoretic terms, we are interested in finding a set M of edges of G such that the following conditions hold:

(i) M is a *matching* in G,

(ii) For each edge in M, one end point is labeled as a transmitter and the other is labeled as a receiver, and

(iii) If $\{x, y\}$ is an edge in M with x as the transmitter, then x is *not* adjacent to any receiver in M other than y. (We refer to y as the **private receiver** of x).

A set M of edges which satisfies the above conditions is called a **Transmitter-Receiver Matching** (or **TR-Matching**)[1] [7, 16]. A transmitting site v and its private receiver v' constitute a **TR-pair** denoted by (v, v'). The cardinality of M is referred to as the **size** of the matching.

In general, the problem of finding a maximum TR-matching (TRM) is known to be NP-complete [7, 16]. In fact, it has been shown to be NP-complete for graphs with maximum node degree 3 [16] and for bipartite graphs [7]. Some efficiently solvable versions of TRM have also been identified. Linear time algorithms for the case when G is a tree and for the case when G is a partial 2-tree are presented in [8] and [6] respectively. In [15] an $O(n^2)$ time algorithm is given for the case when G is an interval graph.

In this paper, we study the complexity of TRM for **chordal graphs**. A graph is **chordal** (or **triangulated**) if every cycle of length 4 or more has a chord [10]. Our results relate the complexity of TRM to the structure of **clique graphs**[2] of chordal graphs. Specifically, we show that the TRM problem is NP-complete for chordal graphs in general, but is solvable in linear time for a subset of chordal graphs, which we call *AC graphs*. These are graphs whose clique graphs are *acyclic*. In proving these results we also establish a number of interesting properties of AC graphs which may be useful for solving other problems.

The remainder of this paper is organized as follows. Section 1 presents the NP-completeness proof. Section 2 presents an efficient algorithm for the special class of chordal graphs mentioned above. Section 3 discusses a linear time implementation of the algorithm presented in Section 2. In the interest of brevity, most proofs are omitted. In some cases, proof sketches are provided. Detailed proofs are available in [4].

---

[1] Also referred to as the **Maximum Open-Irredundant Set problem** [6, 8].

[2] Clique graphs will be defined in Section 2.

2. COMPLEXITY OF TRM FOR CHORDAL GRAPHS

Our proof of the NP-completeness of TRM for chordal graphs uses a reduction from the following special case of TRM which was shown to be NP-complete in [7, page-11].

Problem P1 (TRM for Bipartite Graphs with Preassignment)
Instance: Bipartite graph $G(T,R,E)$, integer K.
Question: Does G have a TR-matching of size K or more in which all transmitters are from the set T and all receivers are from the set R? (A TR-matching where all transmitters are from T and all receivers are from R is **consistent** with the preassignment).

A trivial consequence of the NP-completeness result in [7] is that P1 is NP-complete even when $K \geq 2$. Now consider the following problem:

Problem P2
Instance: A chordal graph $G(V,E)$, integer $K \geq 2$.
Question: Does G have a TR-matching of size K or more?

The NP-completeness of TRM for chordal graphs follows by *restriction* [11] from:

Lemma 1.1: P2 is NP-complete.

Proof (sketch): Membership in **NP** is trivial. To prove that P2 is NP-hard, the restricted version of P1 where $K \geq 2$ is reduced to P2. Given a bipartite graph $G(T,R,E)$ and integer K, we construct a (chordal) graph $G_1(V_1, E_1)$ and integer K_1 as follows: $V_1 = T \cup R$, $E_1 = E \cup E'$, and $K_1 = K$ where $E' = \{\{t_i, t_j\} : \text{for } t_i, t_j \in T, t_i \neq t_j\}$. Note that the edges of E' connect together all of the nodes in T as a clique. Thus, the subgraph of G_1 induced on T is a clique while that induced on R is an independent set. Hence G_1 is a **split graph**, and consequently a chordal graph [10].

Obviously, any TR-matching of size K for P1 is a valid TR-matching of size $K_1 = K$ for G_1. Now suppose that there is a TR-matching M_1 of size K_1 for G_1. It is possible to show that since $K \geq 2$, M_1 cannot contain both a transmitter and a receiver from T. It then follows that in any valid TR-matching of size K_1 for G_1, the set T contains only transmitters (and hence R contains only receivers) or only receivers (and R contains only transmitters). If T contains only receivers then we can obviously "reverse" the TR-matching, making all chosen receivers to be transmitters and all transmitters to be receivers. Thus, we can obtain a TR-matching that is consistent with the preassignment. Therefore, G has a valid TR-matching of size K if and only if G_1 does. ■

The following are immediate consequences of Lemma 1.1.

Theorem 1.2: TRM is NP-complete for chordal graphs.

Corollary 1.3: TRM is NP-complete for split graphs.

3. CHORDAL GRAPHS WITH ACYCLIC CLIQUE GRAPHS

In this section, we present an efficient algorithm for TRM for a subclass of chordal graphs. In order to define this subclass, we need to introduce the notion of the *clique graph* of an undirected graph. Let $G(V,E)$ be an undirected graph. A **clique** of G is a *maximal* set of mutually adjacent vertices. Let $\mathbf{C} = \{C_1, C_2, \cdots, C_m\}$ denote the set of cliques of G. The **clique graph** of G, denoted by $G_c(V_c, E_c)$, is obtained as follows. The nodes in V_c are in one-to-one correspondence with the cliques of $\mathbf{C}$. We let $V_c = \{c_1, c_2, \cdots, c_m\}$. For every $c_i, c_j \in V_c$, the edge $\{c_i, c_j\}$ is in E_c if and only if the corresponding cliques C_i and C_j have a non-empty intersection (i.e., C_i and C_j share at least one node). Several interesting properties of clique graphs of chordal graphs have been established in [3]. These graphs have applications in knowledge-based systems [5, 12], solving systems of linear equations [13, 14], and database systems [2].

We show in this section that TRM is efficiently solvable for graphs whose clique graphs are *acyclic*. As we will show, this class of graphs (referred to as **AC graphs**) is a subset of chordal graphs. Throughout Sections 2 and 3, $G(V,E)$ denotes an AC graph and $G_c(V_c, E_c)$ denotes its clique graph. For simplicity, we will assume that G is connected, and consequently that every clique contains at least two nodes. The extension of our result to disconnected graphs is straight-forward. We begin by stating some simple properties of AC graphs.

Lemma 2.1: Any node of an AC graph belongs to at most two cliques.
Lemma 2.2: Any induced subgraph of an AC graph is also an AC graph.

In order to present other properties of AC graphs, we need some definitions. A **leaf node** of G_c is a node whose degree is at most 1. Since G_c is acyclic, it clearly contains a leaf node. The clique of G corresponding to a leaf node of G_c is called a **leaf clique**. Thus, a leaf clique of G has a non-empty intersection with at most one other clique of G. A node v of G is a **simplicial node** if the set of nodes adjacent to v induces a complete subgraph[3] of G. It is easy to see that a node is simplicial if and only if it belongs to exactly one clique. Our next lemma shows a relationship between leaf cliques and simplicial nodes.

Lemma 2.3: Every leaf clique contains at least one simplicial node.

---

[3] We use the term "complete subgraph" here in preference to "clique" because the latter term is used throughout the paper to denote a *maximal* set of mutually adjacent vertices.

It is well known that a graph G is chordal if and only if there is an ordering $v_1,v_2, \cdots ,v_n$ of the nodes such that for all $1 \leq i \leq n$, v_i is simplicial in the subgraph of G induced on $v_i, \cdots ,v_n$ [10]. Such an ordering is called a **perfect elimination ordering (PEO)**. Using the notion of a PEO and Lemma 2.3 it is easy to prove:

Lemma 2.4: Every AC graph is chordal.

We now prove a key lemma which forms the basis for our algorithm.

Lemma 2.5: For any leaf clique C_l of an AC graph G, there exists a maximum TR-matching of G which contains a TR-pair in C_l. Furthermore, that TR-pair can be made up of two simplicial nodes of C_l, if they exist; otherwise, it can be made up of a simplicial node of C_l and one other node in C_l.

Proof (sketch): Let E_{TR} be the set of edges in some maximum TR-matching of G. Without loss of generality, we can assume that E_{TR} does not contain an edge in C_l, and that $C_l \cap C_p \neq \emptyset$ for some clique $C_p \neq C_l$. It is then possible to show that E_{TR} must contain a TR-pair, say (w,z), from C_p. Let x be a simplicial node of C_l and y be another, simplicial if possible, node in $C_l - \{x\}$ (by Lemma 2.3 x exists; further, since each clique has at least 2 nodes, y also exists). The proof is completed by showing that the TR-pair (w,z) in E_{TR} can be replaced by either (x,y) or (y,x) without affecting the validity of E_{TR}. ∎

We are now ready to discuss our algorithm to find maximum TR-matchings in AC graphs. The basic idea is to successively choose a TR-pair from a leaf clique, and to remove from the graph all nodes that cannot participate in a TR-matching with the chosen pair. We note that the algorithm finds a largest set of **unoriented** pairs (i.e., pairs in which we have not determined which node serves as the transmitter) and then orients them. We use $<x,y>$ to denote an unoriented TR-pair. Our algorithm (AlgTRM) is as follows:

INPUT: An AC graph $G(V,E)$.

OUTPUT: A maximum TR-matching E_{TR} for G.

METHOD: Construct G_c from G. Until all nodes have been removed from G_c, process a leaf clique. To process a leaf clique C_l, a TR-pair $<x,y>$ is formed by finding a simplicial node x and another (simplicial, if possible) node y in C_l. If y is also in the clique C_p which intersects C_l, then C_p cannot have a TR-pair of its own. It can, however, have any number of single endpoints from TR-pairs in other intersecting cliques. Hence, all but one node for each of the other intersecting cliques are removed from G. Next, c_l is removed from G_c and C_l is removed from G. Finally, the TR-pairs are oriented. The details are given in Figure 2.1.

A few comments about Figure 2.1 are in order. First, C_p will fail to exist when C_l is the last clique to be processed. Clearly, an implementation can check for this possibility in step 5(b). Second, c_p is removed from G_c in step 5(c)(v) because at that time C_p has been absorbed by another clique (i.e., it is no longer *maximal*).

Since the main thrust of this section is the optimality of the size of E_{TR}, the details of the orientation step are omitted in Figure 2.1. Briefly, the induced subgraph, G_a, of G on the nodes in E_{TR} *without* the edges in E_{TR} is constructed. The orientation algorithm uses the easily verified facts that nodes in the same connected component (CC) of G_a must have the same orientation (i.e., all are transmitters or all are receivers) and that if $<x,y> \in E_{TR}$ then the CCs containing x and y must have opposite orientations. Thus, this step is similar to 2-coloring a bipartite graph and can therefore be implemented in $O(|V| + |E|)$ time.

```
1. $\mathbf{C} \leftarrow \{C_1, C_2, ..., C_m\}$ (the maximal cliques of $G$)
2. Construct $G_c(V_c, E_c)$ (the clique graph of $G$)
3. $E_{TR} \leftarrow \emptyset$
4. for each clique $C_i$ $PossiblePair(C_i) \leftarrow$ TRUE
5. while $G_c$ is not empty do
     a. $c_l \leftarrow$ a leaf node of $G_c$; $C_l \leftarrow$ the corresponding leaf clique of $G$
     b. $C_p \leftarrow$ the clique that intersects $C_l$
     c. if $PossiblePair(C_l)$ then
          i. $x \leftarrow$ a simplicial node in $C_l$
          ii. if $C_l - \{x\}$ has a simplicial node then
                      $y \leftarrow$ a simplicial node in $C_l - \{x\}$
                      $P \leftarrow \emptyset$
              else
                      $y \leftarrow$ a node in $C_l - \{x\}$
                      if $PossiblePair(C_p)$ then
                              $PossiblePair(C_p) \leftarrow$ FALSE
                              $P \leftarrow \{$ all nodes in $C_p$ except one from each
                                        $C_i \cap C_p, i \neq l$ and $i \neq p\}$
                      else $P \leftarrow \emptyset$
          iii. Add $<x,y>$ to $E_{TR}$
          iv. Delete $c_l$ from $G_c$ and $\{C_l\} \cup P$ from $G$
          v. if $|C_p - P| = 1$ then Delete $c_p$ from $G_c$
        else
          vi. Delete $c_l$ from $G_c$
          vii. Delete the simplicial nodes of $C_l$ from $G$
6. Orient the edges of $E_{TR}$ (See above text)
```

Figure 2.1: Details of AlgTRM

It is not difficult to prove (by induction) that the set of unoriented edges produced in step 5 of the above algorithm is a valid set and that step 6 will correctly orient them. So, we will only prove the following:

Theorem 2.6: The set E_{TR} produced in step 5 of AlgTRM has maximum cardinality.

Proof (sketch): The proof is in two parts.

Part 1: This part proves that at the completion of each iteration of the loop, there is an optimal TR-matching containing the edges chosen so far.

The proof is by induction on k, the number of edges added to E_{TR} so far. The base case ($k = 1$) is Lemma 2.5. For the induction step (the $k + 1^{st}$ edge added to E_{TR}), let $e_1, \cdots, e_k$ be the edges chosen previously by AlgTRM and the next edge chosen be e_{k+1}. Let OPT be an optimal TR-matching that includes the edges $e_1, \cdots, e_k$. We prove (by a somewhat lengthy case analysis) that we can replace one of the edges in $OPT - \{e_1, \cdots, e_k\}$ by e_{k+1} without affecting the validity of OPT.

Part 2: This part proves that at the end of step 5 no proper superset of E_{TR} is a valid TR-matching. The proof is again by a case analysis similar to that in Part 1. ■

4. A LINEAR-TIME IMPLEMENTATION OF AlgTRM

In this section we discuss some implementation details of AlgTRM that provide an $O(|V|+|E|)$ bound on the running time for the algorithm. To begin, we examine the task of producing a clique graph for an AC graph. For this, an adapted version of a maximal clique algorithm in [10] can be used to produce: (1) the set of maximal cliques numbered from 1 to $|\mathbf{C}|$ and (2) an indication of the no more than two cliques that each node of G belongs to. This modified maximal clique algorithm runs in $O(|V|+|E|)$ time. Once we have completed the maximal clique process, we have the $|\mathbf{C}|$ nodes of G_c (i.e., the set of cliques of G) and only need to determine the adjacency list for each node. Due to space limitations, we omit the description of the method used to construct the adjacency lists in linear time. Suffice it to say that it is a modified version of a solution to problem 2.12 on page 71 of [1].

Using this method, the adjacency list and degree count for each clique can be obtained in linear time using $O(|\mathbf{C}|^2 + |V|)$ space[4]. Since the number of cliques in a (finite) chordal graph is bounded above by the number of vertices in the graph [9], the space complexity is $O(|V|^2)$. Once the adjacency lists have been constructed, the extra $O(|V|^2)$ storage is no longer needed. It is used only to prevent duplication of adjacency information.

The data structures needed at the beginning of step 5 (Figure 2.1) are:

1. A list of the nodes in G in random access storage. Along with each node, we maintain an adjacency list. The two representations of each edge are interlinked so that we can easily delete edges from G.
2. For each node in G, a clique membership list.

---

[4] The algorithm will access only $O(|V|)$ of these locations.

3. A list of clique names in random access storage. Along with each clique name C_i we keep its members, an adjacency list, a degree count, and $PossiblePair(C_i)$. Again, the two representations of each edge are interlinked so that we can easily delete edges from G_c.
4. A linked list of the leaf cliques. Initially this will contain all cliques whose adjacency lists are either nil or have just one clique in them. Cliques will be added to the leaf clique list when their degree counts are decremented to 1. Note that this list will become empty only when G_c becomes empty, since G_c is acyclic.

We are now ready to to prove:

Theorem 3.1: AlgTRM has $O(|V|+|E|)$ time complexity.

Proof (sketch): As mentioned above, finding the maximal cliques and constructing the clique graph of G can be done in $O(|V|+|E|)$ time. Steps 3 and 4 can easily be done in $O(|\mathrm{C}|)=O(|V|)$ time. As mentioned earlier, the orientation of edges (step 6) can be done in $O(|V|+|E|)$ time. Consequently, we are left with the complexity of the while loop in step 5. This will be analyzed as three disjoint parts: the accumulated time necessary to execute all statements that require finding simplicial nodes (steps 5(c)(i), 5(c)(ii), and 5(c)(vii)), the accumulated time necessary to execute the statement $P \leftarrow$ {all nodes ... } (in step 5(c)(ii)), and the accumulated time necessary to delete nodes and edges from G and G_c (steps 5(c)(iv), 5(c)(v), 5(c)(vi), and 5(c)(vii)). It is straight-forward to show that the time required for all other parts of the loop is linear.

Recall that a node is simplicial if and only if it is contained in exactly one clique. Thus, given a clique we can determine if one or two nodes in the clique are simplicial by checking, for each node, its clique membership list. This check for membership in more than one clique will take only a constant amount of time for each node. Since, after finding one or two simplicial nodes of a clique C_l, we remove C_l from G before the next iteration of the loop, each clique is searched at most once for simplicial nodes. Thus, because each node appears in no more than two cliques, the accumulated time necessary to execute all statements that search for simplicial nodes (steps 5(c)(i), 5(c)(ii), and 5(c)(vii)) is $O(|V|)$.

It is easy to verify that each execution of the statement $P \leftarrow$ {all nodes ... } (in step 5(c)(ii)) for some clique C_p, needs only $O(|C_p|)$ time. Since this statement is executed only once for each clique C_p, the accumulated time complexity is $O(|V|)$.

The deletion of nodes in from G and G_c (steps 5(c)(iv), 5(c)(v), 5(c)(vi), and 5(c)(vii)) can be done in $O(|V|+|E|)$ using standard techniques.

Summing all the above complexities, we arrive at a time complexity of $O(|V|+|E|)$. ∎

We conclude with a few observations. First, note that AlgTRM successively picks simplicial nodes as endpoints for the edges of E_{TR} in a manner similar to a perfect elimination ordering (PEO). It is not difficult to construct an example to show that an approach that considers the nodes of G in an arbitrary PEO will *not* lead to a maximum TR-matching.

Second, we showed that AC graphs are a subset of chordal graphs. Interval graphs are also a subset of chordal graphs. However, the class of AC graphs neither contains nor is contained in the class of interval graphs.

Finally, it is easy to determine whether or not a given graph is an AC graph. For this, we first verify chordality in linear time using standard techniques [14,17]. Provided that the graph is chordal, we next find the maximal cliques using the modified version of the algorithm in [10] mentioned earlier. Finally we check to see if any node of G belongs to three or more cliques. If a node belongs to three cliques, then the corresponding clique graph will clearly have a cycle in it. Otherwise, we can conclude that the given graph is an AC graph. The reason for this is that it can be shown that if there is a cycle in G_c then there is a node in G that belongs to at least three cliques. Thus, given an arbitrary graph $G(V,E)$, we can determine in $O(|V|+|E|)$ time whether or not the graph is an AC graph.

5. REFERENCES

[1] A.V.Aho, J.E.Hopcroft and J.D.Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley Publishing Company, Reading Massachusetts, 1974.

[2] C.Beeri, R.Fagin, D.Maier and M.Yannakakis, "On the desirability of acyclic database schemes", Journal of the Association for Computing Machinery, Vol. 30, No. 3, July 1983, pp 479-513.

[3] J.R.S.Blair, R.E.England and M.G.Thomason, "Cliques and their Separators in Triangulated Graphs", Technical Report-73-88, Department of Computer Science, University of Tennessee, Knoxville, August 1988.

[4] J.R.S.Blair and S.S.Ravi, "The Complexity of TR-Matching for Chordal Graphs", Tech. Report 88-25, Dept. of Computer Science, SUNY at Albany, Dec. 1988.

[5] G.F.Cooper, "Expert Systems Based on Belief Networks -- Current Research Directions", Technical Report, Medical Computer Science Group, Stanford University, August 1987.

[6] C.J.Colburn and A.Proskurowski, "Concurrent Transmissions in Broadcast Networks", Proc. 11th ICALP, 1984 (Lecture Notes in Computer Science, Vol 172, Springer-Verlag, 1984), pp 128-136.

[7] S.Even, O.Goldreich, S.Moran and P.Tong, "On the NP-completeness of Certain Network-Testing Problems", Networks, Vol. 14, No. 1, Spring 1984, pp 1-24.

[8] A.M.Farley and A.Proskurowski, "On Computing the Open Irredundance Number of a Tree", Technical Report UO-CIS-TR-83-4, Department of Computer and Information Science, University of Oregon, 1983.

[9] F.Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph", SIAM Journal of Computing, Vol. 3, 1974, pp 110-112.

[10] M.C.Golumbic, "Algorithmic Graph Theory and Perfect Graphs", Academic Press, New York, NY, 1980.

[11] M.R.Garey and D.S.Johnson, "Computers and Intractability: A Guide to the Theory of NP-completeness", W.H.Freeman and Co., San Francisco, CA, 1979.

[12] S.L.Lauritzen and D.J.Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems", J. Royal Stat. Soc. B, Vol. 50, No. 2, 1988, pp 157-224.

[13] T.Ohtsuki, L.K.Cheung and T.Fujisawa, "Minimal triangulation of a graph and optimal pivoting order in a sparse matrix", J. Math. Anal. Appl., Vol. 54, 1976, pp 622-633.

[14] D.J.Rose, R.E.Tarjan and G.S.Lueker, "Algorithmic aspects of vertex elimination on graphs", SIAM Journal of Computing, Vol. 5, No. 2, June 1976, pp 266-283.

[15] J.Spinrad, "Private Communication". April 1989.

[16] L.J.Stockmyer and V.V.Vazirani, "NP-Completeness of some Generalizations of the Maximum Matching Problem", Information Processing Letters, Vol. 15, No.1, August 1982, pp 14-19.

[17] R.E.Tarjan and M.Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce hypergraphs", SIAM Journal of Computing, Vol. 13, No. 3, August 1984, pp 566-579.

Generation of Maximum Independent Sets of a Bipartite Graph and Maximum Cliques of a Circular-Arc Graph*

Toshinobu KASHIWABARA and Sumio MASUDA
Department of Information and Computer Sciences
Osaka University, Toyonaka, Osaka 560, Japan

Kazuo NAKAJIMA
Electrical Engineering Department, Institute for Advanced Computer Studies
and Systems Research Center
University of Maryland, College Park, Maryland 20742

Toshio FUJISAWA †

ABSTRACT

We present an efficient algorithm for generating all maximum independent sets of a bipartite graph. Its time complexity is $O(n^{2.5}+(output\ size))$, where n is the number of vertices of a given graph. As its application, we develop an algorithm for generating all maximum cliques of a circular-arc graph. When the graph is given in the form of a family of n arcs on a circle, this algorithm runs in $O(n^{3.5}+(output\ size))$ time.

1. INTRODUCTION

Among various types of combinatorial problems, such as decision problems to determine whether a given structure has specified properties and optimization problems to obtain a best possible structure based on some criteria, generation problems to report all structures that satisfy specified properties without duplication give rise to important applications. In particular, problems of generating all *maximal* or *maximum* sets of elements with specified properties for graphs have widely been investigated because of their practical significance [4,10-15]. (A set of elements is called maximal if the addition of a new element to the set destroys the properties the original elements satisfy. It is called maximum if its cardinality is the largest among all maximal sets.)

For general graphs, the best known algorithms for generating all maximal independent sets (and consequently, maximal cliques) [4,15] and all cutsets [14] require $O(n \cdot m \cdot \alpha)$ and $O((n+m) \cdot \alpha)$ time, respectively, where n, m and α are the numbers of vertices, edges and output sets, respectively. Recently, for several restricted classes of graphs, faster algorithms have been developed. For the generation of maximal independent sets, Leung [10] presented $O(n^2+\gamma)$ time algorithms for interval graphs and circular-arc graphs and an $O((n+m) \cdot \alpha)$ time algorithm for chordal graphs, where γ is the output size, that is, the total sum of sizes of the output sets. Later, Masuda *et al.* [12] developed $O(n \log n+\gamma)$ time algorithms for the first two classes. They also showed that all *maximum* independent sets can be found in $O(n \log n+\gamma)$ time for interval graphs and in $O(n^2+\gamma)$ time for circular-arc graphs. For maximum clique generation, Rotem and Urrutia [13] first presented an $O(n^2+\gamma)$ time algorithm for overlap graphs, and later Masuda *et al.* [11] developed an $O(n\log n+m+\gamma)$ time algorithm.

---

* This work was supported in part by the National Science Foundation Grants MIP-84-51510 and CDR-88-03012 (Engineering Research Centers Program) and in part by a grant from AT&T.

† T. Fujisawa was with the Department of Information and Computer Sciences, Osaka University, Toyonaka, Osaka 560, Japan. He deceased on December 15, 1988.

The time complexities of the generation algorithms for all but one of the above mentioned restricted classes of graphs are the sum of a polynomial function of the input size and a linear function of the output size. We call such generation algorithms *pseudo-linear time* algorithms. In this paper we consider a new problem which admits a pseudo-linear time generation algorithm, that is, the problem of generating all *closed ancestor sets* (defined in Section 2) of an acyclic directed graph. As an application of this algorithm, we present two other pseudo-linear time generation algorithms. The first one produces all maximum independent sets of a bipartite graph in $O(n^{2.5}+\gamma)$ time. The second algorithm generates all maximum cliques of a circular-arc graph in $O(n^{3.5}+\gamma)$ time.

2. DEFINITIONS

We begin by the definitions of several terms on undirected graphs. Let $G = (V, E)$ be an undirected graph. If $(u, v) \in E$, we say that u and v are *adjacent* and are *incident* on the edge (u, v). If $(u, v) \notin E$, they are said to be *independent*. A set of vertices $S \subseteq V$ is called a *clique* (resp., *independent set*) of G if its vertices are pairwise adjacent (resp., independent). A clique (resp., independent set) with the maximum cardinality is called a *maximum clique* (resp., *maximum independent set*, which may be abbreviated as MIS). A set of edges $M \subseteq E$ is called a *matching* of G if no two edges in it have a common endpoint. If a vertex is incident on some edge in M, it is said to be *covered* by M; otherwise it is said to be *exposed*. A matching is called a *perfect matching* if it covers all the vertices of G. A *path* is a sequence of vertices $[v_1, v_2, \ldots, v_k]$ such that $(v_i, v_{i+1}) \in E$ for $i = 1,2,\ldots,k-1$. An *alternating path* with respect to M is a path that contains edges in M and edges not in M alternately (the words "with respect to M" may be omitted if no confusion arises). A graph (V', E') is called a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. A subgraph (V', E') of G is called the *induced subgraph* defined on V' and may be denoted by $G(V')$ if $E' = \{(u, v) \in E \mid u, v \in V'\}$.

We now define some terms on directed graphs (digraphs, for short). Let $\vec{G} = (V, \vec{E})$ be a digraph. Such terms as adjacent vertices, independent set, clique, matching, and induced subgraph of a digraph are defined for its undirected version by ignoring edge directions. A directed edge from vertex u to vertex v is denoted by $(u \to v)$. This edge is an *outgoing edge* of u and an *incoming edge* of v. For a vertex $v \in V$, the number of its incoming (resp., outgoing) edges is called the *indegree* (resp., *outdegree*) and are denoted by $indeg_{\vec{G}}(v)$ (resp., $outdeg_{\vec{G}}(v)$). If a vertex has indegree 0, it is called a *source*. A *directed path* from vertex u to vertex w is a sequence of vertices $[u = v_1, v_2, \ldots, v_k = w]$ such that $(v_i \to v_{i+1}) \in \vec{E}$ for $i = 1,2,\ldots,k-1$. If $u = w$, the path is called a *directed cycle*. A digraph is said to be *strongly connected* if for any pair of vertices u and v, there is a directed path from u to v and vice versa. A *strongly connected component* of digraph $\vec{G} = (V, \vec{E})$ is a strongly connected induced subgraph $\vec{G}(U)$ such that for any $v \in V-U$, $\vec{G}(U \cup \{v\})$ is not strongly connected.

A digraph with no directed cycles is said to be *acyclic*. It is obvious that any acyclic digraph has at least one source. In an acyclic digraph $\vec{G} = (V, \vec{E})$, a vertex u is called an *ancestor* (resp., *descendent*) of vertex v if either $u = v$ or there exists a directed path from u to v (resp., from v to u). A subset S of V is called a *closed ancestor set* (abbreviated as *CA-set*) of $\vec{G}$ if, for any $v \in S$, all ancestors of v are contained in S. An example is given in Figure 1. An edge $(u \to v) \in \vec{E}$ is called a *redundant edge* of $\vec{G}$ if after its elimination, u is still an ancestor of v. A graph $\vec{H}$ is called a *transitive reduction* of $\vec{G}$ if (i) $\vec{H}$ has the same vertex set V and the same ancestor-descendant relationships as $\vec{G}$ has, and (ii) there is no graph with the same vertex set but fewer edges than $\vec{H}$ that preserves the ancestor-descendant relationships of $\vec{G}$. It is known that, for any acyclic digraph $\vec{G}$, its transitive reduction is unique and can be constructed by removing all redundant edges from $\vec{G}$ [1]. Figure 2 depicts the transitive reduction of the acyclic digraph of Figure 1.

For convenience, we may use a few additional notation. A clique of cardinality k may be called a k-clique. A set that contains an element v may be called a v-in set. Thus a clique of cardinality k that contains vertex v may be called a v-in k-clique. For a graph or digraph $G = (V, E)$ and a set of vertices $S \subseteq V$, $G-S$ denotes the induced subgraph $G(V-S)$; that is, the one obtained from G by eliminating all vertices in S and their incident edges.

3. CLOSED ANCESTOR SET GENERATION

In this section, we give a pseudo-linear time generation algorithm for the CA-sets of an acyclic digraph. Let $\vec{G} = (V, \vec{E})$ with $|V| = n$ be an acyclic digraph and $\vec{G}'$ be its transitive reduction. Since $\vec{G}'$ preserves the ancestor-descendant relationships among the vertices of V, any CA-set of $\vec{G}$ is a CA-set of $\vec{G}'$ and vise versa.

Given a subgraph $\vec{H} = (V_H, \vec{E}_H)$ of $\vec{G}'$ and a set of vertices S such that $S \subseteq V - V_H$, the procedure CAS($\vec{H}, S$) (to be given below) finds all CA-sets W of $\vec{H}$ without duplication and produces all sets $W \cup S$. Thus the execution of CAS($\vec{G}', \emptyset$) yields all CA-sets of $\vec{G}'$. They are the CA-sets of $\vec{G}$ from the above observation. Our algorithm is described as follows:

Algorithm 1.

```
1.   Let $\vec{G}' = (V, \vec{E}')$ be the transitive reduction of $\vec{G}$.
2.   CAS($\vec{G}', \emptyset$). □
```

We now explain how to construct CAS($\vec{H}, S$). If $\vec{H} = (\emptyset, \emptyset)$, that is, a null graph, the procedure ($\vec{H}, S$) simply produces S. If $\vec{H} \neq (\emptyset, \emptyset)$, it has at least one source. Let v be any source of $\vec{H}$. Among the sets that are to be produced by CAS($\vec{H}, S$), all sets $W \cup S$ that contain v in W are generated by executing CAS($\vec{H} - \{v\}, S \cup \{v\}$) and all sets $W \cup S$ that do not contain v are obtained by CAS($\vec{H}$–{all descendants of v}, S). Thus, we derive the following recursive procedure:

```
Procedure CAS($\vec{H}, S$)
  begin
    if $\vec{H} = (\emptyset, \emptyset)$
        then output $S$.
        else begin
                 $v \leftarrow$ a source of $\vec{H}$.
                 CAS($\vec{H} - \{v\}, S \cup \{v\}$).
                 CAS($\vec{H}$–{all descendants of $v$}, $S$).
             end
  end □
```

In the remainder of this section, we analyze the time complexity of Algorithm 1. The transitive reduction of $\vec{G}$, as well as the "transitive closure" of $\vec{G}$, can be computed by matrix multiplication [1]. Using the algorithm in [5], we can thus obtain $\vec{G}'$ in $O(n^{2.49+})$ time. To find a source v of $\vec{H}$ when $\vec{H} \neq (\emptyset, \emptyset)$ in the procedure CAS($\vec{H}, S$), we first determine a topological order of $\vec{G}'$ before we call CAS($\vec{G}', \emptyset$). This can be done in $O(n + |\vec{E}'|)$ time [2]. Then, in each execution of CAS($\cdot, \cdot$), a source of the graph can be found as the vertex with the least index with respect to the topological order.

In order to analyze the time complexity of the other parts of the algorithm, we need a few more definitions and lemmas. A pair of vertices is called an *incomparable pair* (abbreviates as *i-pair*) if neither one of them is an ancestor of the other. For any acyclic digraph $\vec{H} = (V_H, \vec{E}_H)$ with no redundant edges, we denote the numbers of its

CA-sets, v-in CA-sets, and i-pairs, by $N_{CAS}(\vec{H})$, $N_{CAS}^{v}(\vec{H})$, and $N_{ip}(\vec{H})$, respectively.

Lemma 1. $|\vec{E}_H| \le 4N_{ip}(\vec{H}) + |V_H|$.
Proof. We will assign each edge to either a vertex or an i-pair. Let $\tilde{H} = (V_H, \tilde{E})$ be a subgraph of $\vec{H}$ such that (i) $indeg_{\tilde{H}}(v) \le 1$ and $outdeg_{\tilde{H}}(v) \le 1$ for every $v \in V_H$, and (ii) for any $e \in \vec{E}_H - \tilde{E}$, $(V_H, \tilde{E} \cup \{e\})$ does not satisfy condition (i). Let $e = (u \to v)$ be any edge in $\vec{E}_H$. Note that $outdeg_{\tilde{H}}(u) = 1$ or $indeg_{\tilde{H}}(v) = 1$ or both equal one. If $e \in \tilde{E}$, we assign it to vertex v. Otherwise, $e \in \vec{E}_H - \tilde{E}$, and we assign it to an i-pair in the following manner:

Case 1. $outdeg_{\tilde{H}}(u) = 0$.

In this case, $indeg_{\tilde{H}}(v) = 1$. Let w be a vertex such that $(w \to v) \in \tilde{E}$. Since $\vec{H}$ has no redundant edges, $\{u, w\}$ is an i-pair. We assign e to this i-pair.

Case 2. $outdeg_{\tilde{H}}(u) = 1$.

There exists a vertex w $(\ne v)$ such that $(u \to w) \in \tilde{E}$ and $\{v, w\}$ is an i-pair. We assign e to this i-pair.

In the manner described above, every edge $e \in \vec{E}_H$ is assigned to a vertex or an i-pair. Clearly, at most one edge is assigned to every vertex. We show that at most 4 edges are assigned to every i-pair, which completes the proof. Assume that five edges e_1, e_2, e_3, e_4, and e_5 are assigned to an i-pair $\{x, y\}$. Since they are incident on either x or y, at least two of them are incoming edges of x or y, or outgoing edges of x or y. For example, suppose that e_1 and e_2 are incoming edges of x, and let $e_1 = (u \to x)$ and $e_2 = (v \to x)$. Since $e_1 \in \vec{E}_H - \tilde{E}$ and e_1 is assigned to the i-pair $\{x, y\}$, the edge $(u \to y) \in \tilde{E}$ must exist due to Case 2. Likewise, $(v \to y) \in \tilde{E}$. Thus, $indeg_{\tilde{H}}(y) \ge 2$, which contradicts the definition of $\tilde{H}$. The other cases lead to a similar contradiction. □

Lemma 2. $N_{CAS}(\vec{H}) \ge N_{ip}(\vec{H}) + |V_H|$.
Proof. For any $v \in V_H$, $\{v$'s ancestors$\}$ is a CA-set. Similarly, for any i-pair $\{u, v\}$ of $\vec{H}$, $\{u$'s ancestors$\} \cup \{v$'s ancestors$\}$ forms a CA-set. It is clear from the definition of i-pairs that these $|V_H| + N_{ip}(\vec{H})$ sets are all distinct, and the lemma follows. □

Lemma 3. For any source v of $\vec{H}$, $|V_H| + |\vec{E}_H| \le 4N_{CAS}^{v}(\vec{H})$.
Proof. If $|V_H| = 1$, the lemma trivially holds. Suppose that $|V_H| \ge 2$ and let $\vec{E}_1$ be the set of all edges of graph $\vec{H} - \{v\}$. From Lemma 1, $|\vec{E}_1| \le 4N_{ip}(\vec{H} - \{v\}) + |V_H| - 1$. Since $|\vec{E}_H| - |\vec{E}_1| \le |V_H| - 1$, we have $|\vec{E}_H| \le 4N_{ip}(\vec{H} - \{v\}) + 2|V_H| - 2$. Furthermore, for any CA-set S of $\vec{H} - \{v\}$, $S \cup \{v\}$ is a CA-set of $\vec{H}$, and hence $N_{CAS}^{v}(\vec{H}) \ge N_{CAS}(\vec{H} - \{v\})$. Therefore, by Lemma 2, $4N_{CAS}^{v}(\vec{H}) \ge 4N_{ip}(\vec{H} - \{v\}) + 4|V_H| - 4 \ge |\vec{E}_H| + 2|V_H| - 2 \ge |V_H| + |\vec{E}_H|$. □

Theorem 1. All CA-sets of a given acyclic digraph $\vec{G}$ can be generated in $O(n^{2.49+} + \gamma)$ time, where γ is the output size.
Proof. As mentioned earlier, the construction of $\vec{G}'$ and the determination of a topological order can be done in $O(n^{2.49+})$ and $O(n + |\vec{E}|)$ time, respectively. For a subgraph $\vec{H} = (V_H, \vec{E}_H)$ of $\vec{G}'$ and $S \subseteq V - V_H$, the execution of CAS$(\vec{H}, S)$ i) finds a source of $\vec{H}$, say $s(\vec{H})$, ii) adds $s(\vec{H})$ to S and computes $\vec{H} - \{s(\vec{H})\}$, and iii) computes $\vec{H} - \{$descendants of $s(\vec{H})\}$. All of these can clearly be performed in $O(1 + |V_H| + |\vec{E}_H|)$ time, which may be bounded by $O(N_{CAS}^{s(\vec{H})}(\vec{H}))$ time due to Lemma 3. Therefore, the overall time complexity of the algorithm is $O(n^{2.49+} + \sum_{\vec{H}, S} N_{CAS}^{s(\vec{H})}(\vec{H}))$, where the summation is taken for all $\vec{H}$ and S

computed in the algorithm. The output family of CAS($\vec{H}$, S), $\{W \cup S \mid W$ is a CA-set of $\vec{H}\}$, has $N_{CAS}^{s(\vec{H})}(\vec{H})$ sets that contain $s(\vec{H})$. This implies that $\sum_{\vec{H},S} N_{CAS}^{s(\vec{H})}(\vec{H}) = O(\gamma)$, which completes the proof. □

Remark. If $\vec{G}$ has no redundant edges, its CA-sets can be generated in $O(\gamma)$ time since $n + |\vec{E}| = O(\gamma)$ by Lemma 3. □

4. GENERATION OF MAXIMUM INDEPENDENT SETS OF A BIPARTITE GRAPH

A graph (V, E) is said to be *bipartite* if V can be partitioned into two disjoint independent sets X and Y. Such a partition (X, Y) is called a *bipartition.* To specify the bipartition, the bipartite graph may be denoted by (X, Y, E).

In this section, we consider the problem of generating all MIS's of a bipartite graph $B = (X, Y, E)$. We first reduce this problem to that of finding all MIS's of a bipartite graph that has a perfect matching. We then show that the latter problem can further be reduced to the generation problem of all CA-sets of an acyclic digraph. These reductions will lead to a pseudo-linear time MIS generation algorithm for B.

4.1 First Reduction

Let M be a maximum matching of B. Let X_0 (resp., Y_0) be the set of exposed vertices in X (resp., Y). Let L_X (resp., L_Y) be the union of X_0 (resp., Y_0) and the set of all vertices of B that lie on some alternating path from a vertex in X_0 (resp., Y_0). Due to the maximality of M, $L_X \cap L_Y = \emptyset$. Finally, we define $X' = X - L_X - L_Y$ and $Y' = Y - L_X - L_Y$. An example is shown in Figure 3. Recall that $B(U)$ is the induced subgraph of B defined on a set of vertices U. For simplicity, we denote graph $B(X' \cup Y')$ by B'. Let M' be the intersection of M and the edge set of B'. It is not difficult to see that M' is a perfect matching of B'. The following theorem enables us to reduce the problem of generating all MIS's of B to that of generating all MIS's of B'.

Theorem 2. The family of MIS's of B is equal to the family $\{(L_X \cap X) \cup (L_Y \cap Y) \cup Z \mid Z$ is an MIS of $B'\}$. □

In order to prove this theorem, we need two lemmas. The proof of Lemma 4 is elementary and is left out.

Lemma 4. B has no edge between i) a vertex in $L_X \cap X$ and a vertex in $(L_Y \cap Y) \cup Y'$ nor ii) a vertex in $L_Y \cap Y$ and a vertex in $(L_X \cap X) \cup X'$. □

Theorem 1 in Desler and Hakimi [6] shows that both $(L_X \cap X) \cup \{v \in Y \mid v$ is not adjacent to any vertex in $L_X\}$ and $(L_Y \cap Y) \cup \{v \in X \mid v$ is not adjacent to any vertex in $L_Y\}$ are MIS's of B. Thus, by Lemma 4, we have Lemma 5.

Lemma 5. $(L_X \cap X) \cup (L_Y \cap Y) \cup X'$ and $(L_X \cap X) \cup (L_Y \cap Y) \cup Y'$ are MIS's of B. □

We are now ready to prove Theorem 2.

Proof of Theorem 2. ("$\supseteq$" part) Let Z be an MIS of B'. Since B' has a perfect

matching M', $|Z|=|X'|=|Y'|$. Furthermore, $(L_X \cap X) \cup (L_Y \cap Y) \cup Z$ is an independent set due to Lemma 4, and Z is disjoint from $(L_X \cap X) \cup (L_Y \cap Y)$. Therefore, $(L_X \cap X) \cup (L_Y \cap Y) \cup Z$ is an MIS of B by Lemma 5.
(“$\subseteq$” part) Let D be an MIS of B. Applying Lemma 5 to $B(L_X)$ and $B(L_Y)$, we see that $L_X \cap X$ and $L_Y \cap Y$ are MIS's of $B(L_X)$ and $B(L_Y)$, respectively. Therefore, $|D \cap L_X| \leq |L_X \cap X|$ and $|D \cap L_Y| \leq |L_Y \cap Y|$. Since B' has a perfect matching, X' is an MIS of B' and $|D \cap (X' \cup Y')| \leq |X'|$. By Lemma 5, $|D| = |(L_X \cap X) \cup (L_Y \cap Y) \cup X'| = |L_X \cap X| + |L_Y \cap Y| + |X'|$. Thus, $|D \cap L_X| = |L_X \cap X|$, $|D \cap L_Y| = |L_Y \cap Y|$ and $|D \cap (X' \cup Y')| = |X'|$, and hence $D \cap L_X$, $D \cap L_Y$ and $D \cap (X' \cup Y')$ are MIS's of $B(L_X)$, $B(L_Y)$ and B', respectively.

The proof will be completed by showing that $L_X \cap X$ (resp., $L_Y \cap Y$) is the only MIS of $B(L_X)$ (resp., $B(L_Y)$). We prove the case for L_X (the other case is similarly proved). $B(L_X - X_0)$ has a perfect matching that consists of edges in M. We denote it by M_X. L_X consists of $2 \cdot |L_X \cap Y|$ vertices covered by M_X and $|X_0|$ exposed vertices. Let D_X be any MIS of $B(L_X)$. Since $|D_X| = |L_X \cap X| = |X_0| + |L_X \cap Y|$, D_X contains all vertices in X_0 and exactly one endpoint of every edge in M_X. Let y be any vertex in $L_X \cap Y$ and $[x_0, y_1, x_1, y_2, x_2, \ldots, y_k, x_k, y]$ be an alternating path, where $x_0 \in X_0$, $y_1, \ldots, y_k \in L_X \cap Y$, $x_1, \ldots, x_k \in L_X \cap X$, and $(y_i, x_i) \in M_X$ for $i = 1,2,\ldots,k$. Since $x_0 \in D_X$, $y_1 \notin D_X$. Thus, $x_1 \in D_X$ and $y_2 \notin D_X$. Repeating this argument $k-1$ more times, we know that $y \notin D_X$. Therefore, D_X has no vertex in $L_X \cap Y$, and hence $D_X = L_X \cap X$. □

4.2 Second Reduction

We now show that the generation of all MIS's of B' can be reduced to that of all CA-sets of an acyclic digraph. Recall that B' has a perfect matching M'. For each edge $e = (u, v)$ such that $u \in X'$ and $v \in Y'$, we give a direction from v to u if $e \in M'$, and from u to v if $e \notin M'$. We denote the resultant digraph by $\vec{B}' = (X', Y', \vec{E}')$.

Theorem 3. Let K be the vertex set of a strongly connected component of $\vec{B}'$. For any MIS D of $\vec{B}'$, either i) $K \cap X' \subseteq D$ and $(K \cap Y') \cap D = \emptyset$, or ii) $K \cap Y' \subseteq D$ and $(K \cap X') \cap D = \emptyset$.
Proof. If $|K| = 1$, the theorem trivially holds. If $|K| > 1$, it is clear that $|K| \geq 4$ and that $|K \cap X'| \geq 2$ and $|K \cap Y'| \geq 2$. Let u be any vertex in $K \cap X'$. For any $v \in (K \cap X') - \{u\}$, there are a directed path from u to v and a directed path from v to u. When the edge directions are ignored, the path from u to v is an alternating path with respect to M' and begins with an edge not in M'. Since M' is a perfect matching, any MIS D of $\vec{B}'$ contains exactly one endpoint of each edge in M'. Therefore, by the same reasoning as the one used in the proof of Theorem 2, we can show that if $u \in D$, then $v \in D$ and $K \cap Y' \not\subseteq D$. Applying the same argument to the directed path from v to u, we can also show that if $u \notin D$, then $v \notin D$.

Consequently, if $u \notin D$, then $(K \cap X') \cap D = \emptyset$, and if $u \in D$, then $K \cap X' \subseteq D$ and $K \cap Y' \not\subseteq D$. A similar result can be derived for any vertex in $K \cap Y'$. The theorem immediately follows from these results. □

Theorem 3 shows that, for the vertex set K of every strongly connected component of $\vec{B}'$, $K \cap X'$ and $K \cap Y'$ each may be treated as a group. This motivates us to contract each of them into a single vertex and create an edge directed from the vertex for $K \cap Y'$ to the one for $K \cap X'$. Let $\hat{B} = (\hat{X}, \hat{Y}, \hat{E})$ be the graph obtained from $\vec{B}'$ by performing such contractions for every strongly connected component (multiple edges are replaced by single edges). Thus, the generation of all MIS's of $\vec{B}'$ is reduced to that of all MIS's of $\hat{B}$. Clearly $\hat{B}$ is acyclic. Furthermore, it has a perfect matching $\hat{M}$ that corresponds to M' and that contains all the new edges. Suppose that $\hat{X} = \{x_1, x_2, ..., x_k\}$, $\hat{Y} = \{y_1, y_2, ..., y_k\}$, and $(y_i \rightarrow x_i) \in \hat{M}$ for $i = 1,2,...,k$. Note that each edge in $\hat{E}-\hat{M}$ is directed from a vertex in $\hat{X}$ to a vertex in $\hat{Y}$.

The second reduction is completed by contracting two endpoints of every edge $(y_i \rightarrow x_i) \in \hat{M}$ into a single vertex v_i (self-loops are eliminated). Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the resultant digraph. Since $\hat{B}$ is acyclic, $\tilde{G}$ is also acyclic. An example is shown in Figure 4. We define a function from the family of all the subsets of $\tilde{V}$ to the family of sets of vertices in $\hat{B}$ by $F(S) = \{x_i \in \hat{X} \mid v_i \notin S\} \cup \{y_i \in \hat{Y} \mid v_i \in S\}$ for every $S \subseteq \tilde{V}$. It is obvious that $F(S)$ is a one-to-one function.

Theorem 4. The family of MIS's of $\hat{B}$ is equal to $\{F(S) \mid S$ is a CA-set of $\tilde{G}\}$.
Proof. ("$\supseteq$" part) Assume that there exists a CA-set S of $\tilde{G}$ such that $F(S)$ contains adjacent vertices x_i and y_j of $\hat{B}$. The edge connecting them is not a member of $\hat{M}$, and hence it is directed from x_i to y_j. By the definition of $F(S)$, $v_i \notin S$ and $v_j \in S$. Since $(x_i \rightarrow y_j) \in \hat{E}-\hat{M}$, $(v_i \rightarrow v_j) \in \tilde{E}$. This contradicts the assumption that S is a CA-set. Therefore, $F(S)$ is an independent set of $\hat{B}$. It is clearly maximum since $|F(S)| = |\hat{X}| = |\hat{Y}|$.
("$\subseteq$" part) Let D be an MIS of $\hat{B}$ and let $S = \{v_i \in \tilde{V} \mid y_i \in D\}$. For each $(y_i \rightarrow x_i) \in \hat{M}$, D contains either x_i or y_i. Therefore, $D = F(S)$. Let $(v_j \rightarrow v_i)$ be any edge in $\tilde{E}$ such that $v_i \in S$. Since $y_i \in D$ and $(x_j \rightarrow y_i) \in \hat{E}$, the independence of D implies that $x_j \notin D$, and hence $v_j \in S$. Thus, S is a CA-set of $\tilde{G}$. □

Theorems 2, 3 and 4 lead to the following algorithm for generating all MIS's of a given bipartite graph B.

Algorithm 2.

1. Find a maximum matching M of B. Determine sets L_X, L_Y, X', and Y' with respect to M and determine graph B' with its perfect matching M'.
2. Determine graph $\vec{B}'$ and find all of its strongly connected components. Construct graph $\hat{B}$ and its perfect matching $\hat{M}$.
3. By contracting every edge in $\hat{M}$, construct graph $\tilde{G}$ from $\hat{B}$.
4. Generate all CA-sets of $\tilde{G}$ and find $\{F(S) \mid S$ is a CA-set of $\tilde{G}\}$.
5. For each set $F(S)$, replace each vertex by the original vertices of B and add to them the vertices in $(L_X \cap X) \cup (L_Y \cap Y)$. Generate the resultant vertex sets. □

Step 1 can be done in $O(n^{2.5})$ time by using the Hopcroft-Karp matching algorithm [9], where n is the number of vertices in B. Step 4 can be performed in $O(n^{2.49+}+\gamma)$ time due to Theorem 1, where γ is the output size. Steps 2, 3 and 5 use $O(n^2)$, $O(n^2)$, and $O(\gamma)$ time, respectively. Thus, we have the following theorem:

Theorem 5. All MIS's of a bipartite graph can be generated in $O(n^{2.5}+\gamma)$ time, where n is the number of vertices of the graph and γ the output size. □

5. GENERATION OF MAXIMUM CLIQUES OF A CIRCULAR-ARC GRAPH

For a family A of arcs on a circle, a graph (V, E) is called the *circular-arc graph* for A if there exists a one-to-one correspondence between V and A such that two vertices in V are adjacent if and only if the corresponding arcs in A intersect. For convenience, we use two terms "vertex" and "arc" interchangeably. Moreover, we may call a clique of the circular-arc graph for A simply a clique of A.

In this section, we present a pseudo-linear time generation algorithm for all maximum cliques of a circular-arc graph. We assume that the graph is given in the form of a family of arcs $A = \{a_1, a_2, ..., a_n\}$ such that their endpoints are all distinct and no single arc covers the whole circle. We assume that arc a_i does not contain any arc in $\{a_{i+1}, a_{i+2}, ..., a_n\}$ for $i= 1,2,...,n-1$ (if necessary, the renumbering can be done by sorting the arcs in ascending order of their lengths). Let μ denote the cardinality of a maximum clique of A. For $i= 1,2,...,n$, let $A_i = \{a_i, a_{i+1}, ..., a_n\}$ and let M_i denote the family of all a_i-in μ-cliques of A_i. Note that M_i may be empty.

It is easy to show that $M_i \cap M_j = \varnothing$ for $i \neq j$ and that the family of maximum cliques of A is $\bigcup\limits_{i=1}^{n} M_i$. Therefore, our goal is to find $M_1, M_2, ..., M_n$. In the following, we show that this problem can be reduced to the generation of all MIS's of a bipartite graph. Let p_i^l be one endpoint of arc a_i and p_i^r the other endpoint. We denote by A_i^l (resp., A_i^r) the set of all arcs in A_i that contain p_i^l but not p_i^r (resp., p_i^r but not p_i^l) and by A_i^{lr} the set of all arcs in A_i that contain both p_i^l and p_i^r. Let $B_i = (A_i^l \cup A_i^r, E_i)$ with $E_i = \{(u, v) \mid u,v \in A_i^l \cup A_i^r$, arcs u and v do not intersect$\}$. An example is given in Figure 5. B_i is a bipartite graph with bipartition (A_i^l, A_i^r). Such a bipartite graph was used in [7] to obtain a maximum clique of a circular-arc graph. Let τ_i denote the cardinality of an MIS of B_i. If $\tau_i + |A_i^{lr}| + 1 < \mu$, we know that $M_i = \varnothing$.

Theorem 6. If $\tau_i + |A_i^{lr}| + 1 = \mu$, then $M_i = \{A_i^{lr} \cup \{a_i\} \cup I \mid I$ is an MIS of $B_i\}$.
Proof. ("$\supseteq$" part) By the definition of B_i, any MIS I of B_i is a clique of A_i. Since every arc in $A_i^{lr} \cup \{a_i\}$ intersects all arcs in I, $A_i^{lr} \cup \{a_i\} \cup I$ is an a_i-in μ-clique of A_i.
("$\subseteq$" part) Let C be an a_i-in μ-clique of A_i. $C - \{a_i\} - A_i^{lr}$ is a clique of $A_i^l \cup A_i^r$, and thus it is an independent set of B_i. Since $|C| = \mu$, $|C - \{a_i\} - A_i^{lr}| \geq \mu - 1 - |A_i^{lr}| = \tau_i$. Therefore, $C - \{a_i\} - A_i^{lr}$ is an MIS of B_i. □

For $i= 1,2,...,n$, we first compute τ_i by finding a maximum clique of A_i. Using the algorithm by Apostolico and Hambrusch [3], this can be done in $O(n^2 \log \log n)$ time for each i. μ can be determined as $Max\{\tau_i + 1 + |A_i^{lr}| \mid i= 1,2,...,n\}$. If $\tau_i < \mu - 1 - |A_i^{lr}|$ for some i, we do not proceed further for i. On the other hand, if $\tau_i = \mu - 1 - |A_i^{lr}|$, we find M_i in the following manner: (1) construct graph B_i, (2) generate all MIS's of B_i, and (3) add $\{a_i\} \cup A_i^{lr}$ to each of them. The execution times of the first and third steps for all i are $O(n^3)$ and $O(\gamma)$, respectively. Due to Theorem 5, the second step can be done in $O(n^{2.5} + \gamma_i)$ time for each i, where γ_i is the total size of the MIS's of B_i. Since $\sum\limits_{\tau_i = \mu - 1 - |A_i^{lr}|} \gamma_i = O(\gamma)$, we have the following theorem:

Theorem 7. The generation of all maximum cliques of a circular-arc graph can be done in $O(n^{3.5}+\gamma)$ time, where n is the number of vertices of the graph and γ the output size. □

REFERENCES

[1] A. V. Aho, M. R. Garey and J. D. Ullman, "The Transitive Reduction of a Directed Graph," *SIAM J. Comput.*, vol. 1, pp. 131-137, 1972.

[2] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1972.

[3] A. Apostolico and S. E. Hambrusch, "Finding Maximum Cliques on Circular-Arc Graphs," *Info. Proc. Lett.*, vol. 26, pp. 209-215, 1987.

[4] N. Chiba and T. Nishizeki, "Arboricity and Subgraph Listing Algorithms," *SIAM J. Comput.*, vol. 14, pp. 210-223, 1985.

[5] D. Coppersmith and S. Winograd, "On the Asymptotic Complexity of Matrix Manipulation," *SIAM J. Comput.*, vol. 11, pp. 472-492, 1982.

[6] J. F. Desler and S. L. Hakimi, "On Finding a Maximum Internally Stable Set of a Graph," *Proc. 4th Annual Princeton Conf. on Information Sciences and Systems*, Princeton, NJ, 1970, pp. 459-462.

[7] F. Gavril, "Algorithms on Circular-Arc Graphs," *Networks*, vol. 1, pp. 357-369, 1974.

[8] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, NY, 1980.

[9] J. E. Hopcroft and R. E. Karp, "An $n^{2.5}$ Algorithm for Maximum Matching in Bipartite Graphs," *SIAM J. Comput.*, vol. 2, pp. 225-231, 1973.

[10] J. Y.-T. Leung, "Fast Algorithms for Generating All Maximal Independent Sets of Interval, Circular-Arc and Chordal Graphs," *J. Algorithms*, vol. 5, pp. 22-35, 1984.

[11] S. Masuda, K. Nakajima, T. Kashiwabara and T. Fujisawa, "Efficient Algorithms for Finding Maximum Cliques of an Overlap Graphs," *Networks*, to appear.

[12] S. Masuda, K. Nakajima, T. Kashiwabara and T. Fujisawa, *Efficient Enumeration of Maximal and Maximum Independent Sets of an Interval Graph and a Circular-Arc Graph*, Technical Report UMIACS-TR-87-33, University of Maryland, College Park, MD, 1987.

[13] D. Rotem and J. Urrutia, "Finding Maximum Cliques in Circle Graphs," *Networks*, vol. 11, pp. 269-278, 1981.

[14] S. Tsukiyama, H. Ariyoshi, I. Shirakawa and H. Ozaki, "An Algorithm to Enumerate All Cutsets of a Graph in Linear Time per Cutset," *J. Assoc. Comput. Mach.*, vol. 27, pp. 619-632, 1980.

[15] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, "A New Algorithm for Generating All the Maximal Independent Sets," *SIAM J. Comput.*, vol. 6, pp. 505-517, 1977.

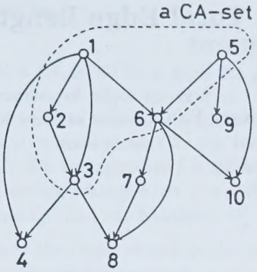

a CA-set
1
2
3
4
5
6
7
8
9
10

Fig. 1. An example of a CA-set.

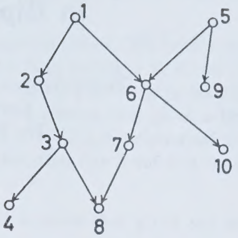

1
2
3
4
5
6
7
8
9
10

Fig. 2. The transitive reduction for the acyclic digraph of Fig. 1.

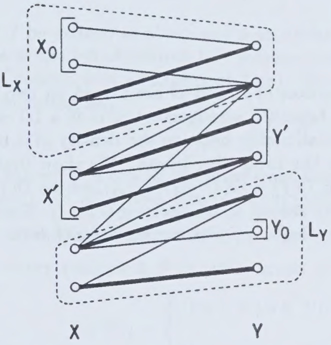

X_0
L_X
Y'
X'
Y_0
L_Y
X
Y

Fig. 3. Sets X_0, Y_0, L_X, L_Y, X' and Y' (the edges of M are drawn in boldface).

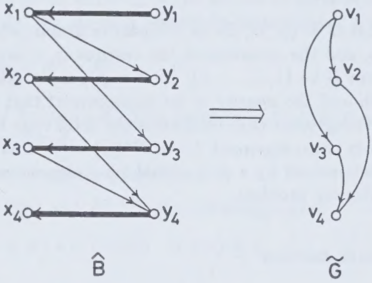

x_1
x_2
x_3
x_4
y_1
y_2
y_3
y_4
$\hat{B}$
v_1
v_2
v_3
v_4
$\tilde{G}$

Fig. 4. Construction of graph $\tilde{G}$.

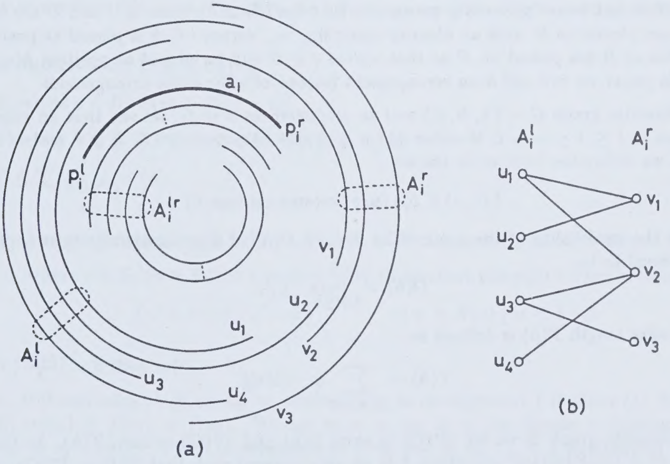

a_i
p_i^r
p_i^l
A_i^{lr}
A_i^r
A_i^l
v_1
u_2
u_1
v_2
u_3
u_4
v_3
(a)
A_i^l
A_i^r
u_1
u_2
u_3
u_4
v_1
v_2
v_3
(b)

Fig. 5. (a) Sets A_i^l, A_i^r and A_i^{lr}. (b) Graph B_i.

Relationship Between Density and Total Edge Length in Bipartite Arrangement

Thang Nguyen Bui, Sing-Ling Lee and Kang-Pu Liou
Computer Science Department
The Pennsylvania State University
University Park, Pa. 16802

Abstract

Let $G = (A, B, E)$ be a bipartite graph, where the vertices of A and B are placed on two parallel rows, and the positions of the vertices in A are fixed. A bipartite arrangement of G is a 1-1 mapping h from B to $\{1, 2, \ldots, n\}$. In this paper we establish a relationship between the density of a bipartite graph and the density of an arrangement that minimizes the total edge length. We show that if f is an arrangement that minimizes the total edge length then $D(f) = O(D^*(G) \log^2 n)$, where $D(f)$ is the density of arrangement f and $D^*(G)$ is the smallest density among all arrangements for G. Since f can be determined by a polynomial time algorithm, we have a $\log^2 n$-approximation algorithm for solving the density problem.

I. Introduction

Let $G = (A, B, E)$ be a bipartite graph with $|A| = |B| = n$, the set A is identified with the set $\{1, 2, \ldots, n\}$. A *bipartite arrangement* of G is a 1-1 mapping h from B to $\{1, 2, \ldots, n\}$. Given a bipartite arrangement h we will place the vertices in A and B on two horizontal lines U and D as follows. Each of these two horizontal lines have markings which are one unit apart and they are labeled from 1 to n (from left to right). Without loss of generality we assume that the left end points of U and D are aligned. The vertices of A are placed on U with an identity mapping, i.e., vertex $i \in A$ is placed at position i on U; and the vertices of B are placed on D so that vertex $v \in B$ will be placed at position $h(v)$ on D. For the rest of this paper we will call h an arrangement instead of a bipartite arrangement.

Given a bipartite graph $G = (A, B, E)$ and an arrangement h of G, we say that an edge $(u, v) \in E$ *crosses column* i, $1 \le i \le n - 1$, if either (1) $u \le i$ and $h(v) > i$ or (2) $u > i$ and $h(v) \le i$. For $1 \le i \le n - 1$ we define the *i-cut* to be the set

$$\{ (u, v) \in E \mid (u, v) \text{ crosses column } i \}$$

and C_i^h to be the cardinality of the i-cut. The *density* $D(h)$ of a given arrangement h of a bipartite graph G is defined to be

$$D(h) = \max_{1 \le i \le n-1} C_i^h,$$

and the total edge length $T(h)$ is defined as

$$T(h) = \sum_{(u,v) \in E} |u - h(v)|.$$

For a given bipartite graph G we let $D^*(G) = \min_h D(h)$ and $T^*(G) = \min_h T(h)$. In this paper we show that $D(f) = O(D^*(G) \log^2 n)$, where f is an arrangement such that $T(f) = T^*(G)$. Since f can be determined by a polynomial time algorithm [BL87], we have a $\log^2 n$-approximation algorithm for solving the density problem. This result has applications on the placement problem that minimizes the channel density in the standard cell layout of integrated circuits [PDS, PK].

II. Definitions

Let $G = (A, B, E)$ be a bipartite graph and h be an arrangement of G. For each $v \in B$ let $C_i^h(v)$ be the number of edges incident to v that cross the i-cut. For each vertex $v \in B$, let $N(v) \subseteq A$ be the set of vertices adjacent to it, and let $T_h(v) = \sum_{u \in N(v)} |u - h(v)|$ be the edge length contributed by vertex v in arrangement h. For the remainder of the paper, we assume that given a bipartite graph $G = (A, B, E)$, arrangement f is an arrangement such that $T(f) = T^*(G)$. A polynomial time algorithm for constructing arrangement f is given in [BL87]. Let k be such that the k-cut is a maximum cut in arrangement f, i.e., cut k satisfies $C_k^f = \max_{1 \leq i \leq n} C_i^f$.

To show the result stated in the introduction it is sufficient to show that given any arrangement g there is a cut i in arrangement g such that

$$C_i^g = \Omega\left(\frac{C_k^f}{\log^2 n}\right).$$

Let f be defined as above and g an arbitrary arrangement of G. Let $L^f \subseteq B$ ($L^g \subseteq B$) be the set of vertices to the left of column k in arrangement f (g) and $R^f \subseteq B$ ($R^g \subseteq B$) be the set of vertices to the right of column k in arrangement f (g). We define set S_1 and S_2 with respect to arrangement f and g in the following way:

$$S_1 = L^f - L^g \text{ and } S_2 = R^f - R^g$$

Arrangement g can be obtained from arrangement f by moving all the vertices in S_1 (S_2) from the left (right) side of column k to the right (left) side of column k.

We start by giving some definitions and prove some elementary properties.

For every vertex $v \in B$ in arrangement f we define $P(v)$ and $R(v)$ in the following way:

$$P'(v) = \begin{cases} |\{\, u > k \mid u \in N(v) \,\}| - |\{\, u \leq k \mid u \in N(v) \,\}| & \text{if } f(v) \leq k, \\ |\{\, u \leq k \mid u \in N(v) \,\}| - |\{\, u > k \mid u \in N(v) \,\}| & \text{if } f(v) > k, \end{cases}$$

$$P(v) = \max(P'(v), 0) \text{ and } R(v) = \frac{|N(v)| - |P'(v)|}{2}.$$

It is easy to see that $P(v)$ and $R(v)$ have the following properties with respect to arrangement f and g:

1) For each vertex $v \in B$, $R(v) \leq C_k^g(v)$.

2) For each vertex $v \in B$, $P(v) + R(v) = C_k^f$.

3) For each vertex $v \in B$, if $P(v) = 0$ then $C_k^f(v) \geq C_k^g(v)$.

4) $C_k^f - C_k^g \leq \sum_{v \in S_1 \cup S_2} P(v)$.

We define the *margin function*, $m\colon S_1 \cup S_2 \to A \cup \{0\}$, in arrangement f as follows:

$m(v) = u$, where $v \in S_1$ ($v \in S_2$), if u is the $\lceil \frac{P(v)}{4} \rceil$-th smallest (largest) vertex in the set

$$\{\, u' \in N(v) \mid u' > k \,\} \qquad (\{\, u' \in N(v) \mid u' \leq k \,\}).$$

Note that if $\lceil \frac{P(v)}{4} \rceil = 0$ then $m(v) = 0$.

Let $v_1, v_2 \in B$ and $m(v_2) > 0$, we say v_1 *dominates* v_2 in arrangement f if either (1) $f(v_2) < f(v_1) \leq m(v_2)$ or (2) $m(v_2) \leq f(v_1) < f(v_2)$. We use $v_1 \propto v_2$ ($v_1 \not\propto v_2$) to denote v_1 dominates (does not dominate) v_2. If $v_1 \propto v_2 \propto \ldots \propto v_t$ in arrangement f we call $v_1, v_2, \ldots, v_t$ a *dominating sequence*. The following lemma specifies the relative positions of a dominating sequence.

Lemma 2.1. Let $v_1, v_2, \ldots, v_t$ be a dominating sequence in arrangement f, then the following properties hold:

1) If $f(v_1) < f(v_2)$ then $\cdots < f(v_{2i+1}) < f(v_{2i-1}) < \cdots < f(v_1) < k \leq f(v_2) < \cdots < f(v_{2i}) < f(v_{2i+2}) < \cdots$.

2) If $f(v_1) > f(v_2)$ then $\cdots > f(v_{2i+1}) > f(v_{2i-1}) > \cdots > f(v_1) \geq k > f(v_2) > \cdots > f(v_{2i}) > f(v_{2i+2}) > \cdots$.

Corollary 2.1. If $v_1, v_2, \ldots, v_t$ is a dominating sequence in arrangement f, then $v_1 \propto v_{2i}, 1 \leq i \leq \lfloor \frac{t}{2} \rfloor$.

We now show how to partition vertices in $S_1 \cup S_2$ into subsets such that each subset forms a dominating sequence. We first construct a directed graph $H = (S_1 \cup S_2, Z_1 \cup Z_2)$, where $Z_1 \subseteq S_2 \times S_1$ and $Z_2 \subseteq S_1 \times S_2$. We only show how to generate edges in Z_1. Edges in Z_2 can be obtained in a similar way.

Let $F_1 = \{ v \in S_1 \mid m(v) > 0 \}$, we partition set F_1 into subsets $X_0, X_1, \ldots, X_t$ and set S_2 into subsets $Y_1, Y_2, \ldots, Y_{t+1}$ as follows.

Let $X_0 = \{ w \in F_1 \mid \forall v \in S_2, v \not\propto w \}$, and

$$X_i = \{ w \in F_1 \mid \forall v \in S_2 - \cup_{j=1}^{i} Y_j, v \not\propto w \}, \qquad \text{for } i = 1, \ldots, t,$$

$$Y_i = \{ v \in S_2 \mid \forall w \in F_1 - \cup_{j=0}^{i-1} X_j, v \propto w \}, \qquad \text{for } i = 1, \ldots, t+1,$$

We construct the edge set Z_1 by the following procedure.

```
$RT = \emptyset$ and $Z_1 = \emptyset$.
For $i = t$ downto 1 Do
    $RT = RT \cup X_i$
    while ($Y_i \neq \emptyset$ and $RT \neq \emptyset$) Do
        Let $w \in RT$ such that $f(w) \geq f(w')$ for $w' \in RT$.
        Arbitrarily choose a vertex $v \in Y_i$, $Z_1 = Z_1 \cup \{(v, w)\}$
        $RT = RT - \{w\}$
        $Y_i = Y_i - \{v\}$
    endwhile
endfor
```

From Lemma 2.1, we know that the directed graph H is acyclic and consists of only vertex disjoint paths. Since $u \propto v$ if (u, v) is an edge in H, each of these paths is a dominating sequence. We call a vertex $v \in S_1 \cup S_2$ a *root* if the indegree of v is zero in graph H. Since H is acyclic, there exists at least one root in H. We define *root set* $Q \subseteq S_1 \cup S_2$ and *non_root set* $W \subseteq S_1 \cup S_2$ from graph H as follows:

$$Q = \{ v \in S_1 \cup S_2 \mid v \text{ is a root} \}$$

$$W = S_1 \cup S_2 - Q.$$

For each root $v \in Q \neq \emptyset$, let

$$D(v) = \{ w \in W \mid w \text{ is a descendant of } v \text{ in } H \},$$

$$\Delta(v) = \sum_{w \in D(v)} P(w).$$

In other words, $\Delta(v)$ is an upper bound for the number of edges which cross cut k in f but do not cross cut k in g for all descendants of v in graph H. Clearly, we also have $\sum_{w \in W} P(w) = \sum_{v \in Q} \Delta(v)$.

We are now ready to give the main idea of the proof of our main result. First observe that $\sum_{w \in W} P(w)$ is an upper bound on the total reduction in the size of cut k caused by vertices in the non-root set W, when we go from arrangement f to arrangement g. Similarly, $\sum_{v \in Q} P(v)$ is an upper bound on the reduction in the size of k caused by vertices in the root set Q. In Sections III and IV, we show that there are cuts k_1 and k_2 in arrangement g such that

$$C^g_{k_1} = \Omega\left(\frac{\sum_{w \in W} P(w)}{\log^2 n}\right) \qquad \text{and} \qquad C^g_{k_2} = \Omega\left(\frac{\sum_{v \in Q} P(v)}{\log^2 n}\right).$$

Since $C^g_k \ge C^f_k - \sum_{v \in Q \cup W} P(v)$, it is not difficult to see that

$$C^f_k = O\left(\max\{C^g_k, C^g_{k_1}, C^g_{k_2}\} \log^2 n\right),$$

which is the main result stated in the introduction.

III. Lower Bound on the Size of Cut k_1 in Arrangement g

In this section we will use the dominating sequences constructed from graph H to find the cut k_1 in arrangement g. Important properties of dominating sequence are given in the following lemmas.

Lemma 3.1. Let $v_i, v_j \in B$ if $f(v_j) < f(v_i) \le k$, $(k < f(v_i) < f(v_j))$, and $P(v_j) > 0$ then $|N(v_i)| \ge P(v_j)$

Corollary 3.1. If $v_i \propto v_j$ then $|N(v_i)| \ge 0.5 \cdot P(v_j)$.

From Lemmas 2.1, 3.1 and Corollaries 2.1, 3.1 we can easily obtain the following lemma.

Lemma 3.2. If $v_1, v_2, \ldots, v_t$ is a dominating sequence in arrangement f then $|N(v_1)| \ge 0.5 \cdot P(v_i)$ for each vertex $v_i, 2 \le i \le t$.

Proof: Without loss of generality assume that $f(v_1) < f(v_2)$ (the other case is proved similarly.) From Lemma 2.1 and Corollary 2.1 we know that (1) $f(v_1) > f(v_{2i+1})$ and (2) $v_1 \propto v_{2i}$, for $1 \le i \le \lfloor \frac{t}{2} \rfloor$. The proof of this lemma can be completed by using Lemma 3.1 and its corollary. ∎

In the next two lemmas we establish the relationship between the degree of a root v and $\Delta(v)$, where $\Delta(v)$ is an upper bound on the number of edges contributed by v's descendants on cut k in arrangement f. Lemma 3.3 shows that if $|N(v)|$ is "large", then $\Delta(v)$ is bounded by $c \log n \cdot |N(v)|$. Lemma 3.4 shows that if $|N(v)|$ is "small", then $\sum_w C^g_k(w)$, w is a descendant of v, is at least $\frac{\Delta(v)}{c \log n}$.

In the proofs of Lemma 3.3 and Lemma 3.4, we use the fact that given any sequence $d_1, d_2, \ldots, d_t$ with $d_i > 0$ and $\sum_{i=1}^t d_i \le n$, we have $\sum_{i=1}^t \frac{d_i}{\sum_{j=1}^i d_j} < \log n + 1$. This can be shown by rewriting the sequence as a harmonic series.

Lemma 3.3. If $v_1, v_2, \ldots, v_t$ is a dominating sequence and $|N(v_1)| \ge \sum_{i=1}^t R(v_i)$ then

$$\sum_{i=2}^{t} P(v_i) = O(\log n |N(v_1)|).$$

Proof: Without loss of generality assume that $f(v_1) < f(v_2)$ (the other case is similar). Let $d_i = |f(v_{i+1}) - f(v_{i-1})|$, where $1 \le i < t$ and $f(v_0) = f(v_1)$. Consider arrangement h_i which is obtained by swapping vertices v_i and v_{i+1} in arrangement f, $1 \le i < t$. Since

$$T_{h_i}(v_{i+1}) - T_f(v_{i+1}) \le -\frac{P(v_{i+1})}{2} \cdot \sum_{j=1}^{i} d_j,$$

$$T_{h_i}(v_i) - T_f(v_i) \le |N(v_i)| \cdot d_i, \quad \text{and}$$

$$T(h_i) - T(f) = (T_{h_i}(v_i) - T_f(v_i)) + (T_{h_i}(v_{i+1}) - T_f(v_{i+1})) \ge 0,$$

We have
$$\frac{P(v_{i+1})}{2} \cdot \sum_{j=1}^{i} d_j \le |N(v_i)| \cdot d_i,$$

$$P(v_{i+1}) \le \frac{d_i}{\sum_{j=1}^{i} d_j} \cdot 2|N(v_i)|$$

Sum the left side and right side of inequalities obtained from arrangement h_i, $1 \leq i < t$, we have

$$\begin{aligned}\sum_{i=2}^{t} P(v_i) &\leq \sum_{i=2}^{t} \frac{d_i}{\sum_{j=1}^{i} d_j} 2|N(v_i)| \\ &= \sum_{i=2}^{t} \frac{d_i}{\sum_{j=1}^{i} d_j} (2P(v_i) + 4R(v_i)) \\ &\leq \sum_{i=2}^{t} \frac{d_i}{\sum_{j=1}^{i} d_j} (4|N(v_1)| + 4R(v_i)), \qquad \text{(Lemma 3.2)} \qquad (*)\end{aligned}$$

By assumption $|N(v_1)| \geq \sum_{l=1}^{t} R(v_l)$ hence

$$\begin{aligned}\sum_{i=2}^{t} P(v_i) &\leq \sum_{i=2}^{t} \frac{d_i}{\sum_{j=1}^{i} d_j} 8|N(v_1)| \\ &= O(|N(v_1)| \log n). \quad \blacksquare\end{aligned}$$

Lemma 3.4. If $v_1, v_2, \ldots, v_t$ is a dominating sequence and $|N(v_1)| < \sum_{i=1}^{t} R(v_i)$ then

$$\sum_{i=1}^{t} C_k^f(v_i) = O\left(\log n \cdot \sum_{i=1}^{t} C_k^g(v_i)\right).$$

Proof: Since $\sum_{i=1}^{t} C_k^f(v_i) = \sum_{i=1}^{t}(P(v_i) + R(v_i))$ and $\sum_{i=1}^{t} C_k^g(v_i) \geq \sum_{i=1}^{t} R(v_i)$, we know that

$$\begin{aligned}\frac{\sum_{i=1}^{t} C_k^f(v_i)}{\sum_{i=1}^{t} C_k^g(v_i)} &\leq \frac{\sum_{i=1}^{t}(P(v_i) + 2R(v_i))}{\sum_{i=1}^{t} R(v_i)} \\ &= 2 + \frac{\sum_{i=1}^{t} P(v_i)}{\sum_{i=1}^{t} R(v_i)} \\ &\leq 2 + \frac{\sum_{i=1}^{t} \frac{d_i}{\sum_{j=1}^{i} d_j}(4|N(v_1)| + 4R(v_i))}{\sum_{i=1}^{t} R(v_i)} \qquad \text{from } (*) \text{ in the previous lemma} \\ &= O(\log n) \qquad (\text{from } |N(v_1)| < \sum_{l=1}^{t} R(v_l)) \quad \blacksquare\end{aligned}$$

For each vertex $v \in Q$, we call v a *k_root vertex* if with respect to cut k in arrangement f and g, it has the following property:

$$\frac{\sum_{w \in D(v)} C_k^f(w)}{\sum_{w \in D(v)} C_k^g(w) + C_k^g(v)} = O(\log n).$$

We call v a *non k_root vertex* if v is not a k_root . Set $Q' \subseteq Q$ is called a *k_root set* if each vertex v, $v \in Q'$, is a k_root vertex. Otherwise, we call set Q' a *non k_root set.* We shall now partition set Q into four subsets Q_1, Q_2, Q_3, and Q_4 such that sets Q_1 and Q_2 are k_root set and Q_4 is a non k_root set. Set Q_3 is not a k_root set but can be treated as a k_root set by virtue of Lemma 3.7.

We give the definitions of sets Q_1 and Q_2 in the following.

$$\begin{aligned}Q_1 &= \{ v \in Q \mid P(v) = 0 \}, \\ Q_2 &= \{ v \in Q \mid |N(v)| < R(v) + \sum_{w \in D(v)} R(w) \}.\end{aligned}$$

Lemma 3.4 immediately implies that Q_2 is a k_root set. We only show that set Q_1 is a k_root sets in the following lemma.

Lemma 3.5. Q_1 and Q_2 are k_root sets.
Proof: From the definition of $D(v)$, we know that there is a dominating sequence consisting of vertex

$v \in Q$ and vertices in set $D(v)$ with vertex v as the root. Without loss of generality we may assume that $v \in Q_1 \cap S_1$. We show that v is a k_root vertex by considering two cases.

Case 1: $|N(v)| \geq R(v) + \sum_{w \in D(v)} R(w)$

Since $P(v) = 0$ and $v \in L^f$, that means v is located to the left of cut k in arrangement f, we know that $C_k^g(v) \geq 0.5 \cdot |N(v)|$, where vertex $v \in R^g$. Lemma 3.3 gives $\sum_{w \in D(v)} P(w) = O(|N(v)| \log n)$. Hence, v is a k_root vertex.

Case 2: $|N(v)| < R(v) + \sum_{w \in D(v)} R(w)$

Since v and $D(v)$ form a dominating sequence, from Lemma 3.4, we know that v is a k_root vertex. This completes the proof of the lemma. ∎

In the following, we define $\overline{X_j}$ and $\overline{Y_j}$, where $0 \leq j \leq l+1$, by grouping X_i and Y_i obtained in Section II. We then use $\overline{X_j}$ and $\overline{Y_j}$ to generate sets Q_3 and Q_4.

(1) $\overline{X_0} = X_0$; $\overline{X_{l+1}} = \emptyset$; $\overline{Y_0} = \emptyset$; $\overline{Y_{l+1}} = Y_{t+1}$;

(2) $\overline{X_j} = \bigcup_{i=a}^{b} X_i$, where $\bigcup_{s=0}^{j-1} \overline{X_s} = \bigcup_{i=0}^{a-1} X_i$, and $a \leq b \leq t$.

$\overline{Y_j} = \bigcup_{i=a}^{b} Y_i$, where $\bigcup_{s=0}^{j-1} \overline{Y_s} = \bigcup_{i=0}^{a-1} Y_i$, and $a \leq b \leq t$.

(3) $\forall \overline{y} \in \overline{Y_j}$, its child must be in $\overline{X_j}$ and $\forall \overline{x} \in \overline{X_j}$, its parent must be in $\overline{Y_j}$

(4) no subsets of $\overline{X_j}$ and Y_j satisfy the above three properties.

Set Q_3 can be partitioned into two subsets Q_3' and Q_3'', such that $Q_3' \subseteq S_1$ and $Q_3'' \subseteq S_2$. We shall show how to construct set Q_3' in the following. Set Q_3'' can be obtained in a similar way.

We use sets $\overline{X_j}$ and $\overline{Y_j}$ to generate set Q_3'. For each set $\overline{Y_j}, 1 \leq j \leq l+1$, let $\overline{y} \in \overline{Y_j}$ be the leftmost vertex in $\overline{Y_j}$ such that $P(\overline{y}) > 0$. Let

$$\overline{Y_j'} = \{\overline{y'} \in \overline{Y_j} \mid f(\overline{y'}) < f(\overline{y})\},$$
$$\overline{X_j'} = \overline{X_j} \cap \{Q - Q_1 \cup Q_2\}, 0 \leq j \leq l+1.$$

We use the following procedure A_CUT to generate set Q_3' and some cuts a_i which will be used later for set Q_4.

```
procedure A_CUT;
    $Q_3' = \emptyset$; $s = 1$;
    for $j = 0$ to $l$ do
        if $|Y_{j+1}'| \geq |X_j'|$ then
            $Q_3' = Q_3' \cup X_j'$
            each vertex $v \in X_j'$ is paired with a vertex $r \in Y_{j+1}'$
        else
            Let $X_j''$ be a subset of $X_j'$ with $|X_j''| = |Y_{j+1}'|$ such that
            if $u \in X_j''$ and $v \in X_j' - X_j''$ then $f(v) > f(u)$ in arrangement $f$;
            $Q_3' = Q_3' \cup X_j''$;
            if $\exists y \in Y_j$ such that $P(y) > 0$ then
                define cut $a_s$ such that $a_s = f(y) - 1$;
                $s = s + 1$;
    endfor
end. { A_CUT }
```

Properties of cut a_j generated in the procedure A_CUT are given in Lemma 3.6 and Lemma 3.7. For each cut a_j in arrangement f, let $A_j = \{\, v \in S_1 \mid 0 < m(v) \leq a_j \,\}$, $B_j = \{\, u \in S_2 \mid f(u) \leq a_j \,\}$, and $\overline{B_j} = \{\, u \in B - S_2 \mid k \leq f(u) < a_j \,\}$.

Lemma 3.6. For each cut a_j, there will be at least $|A_j| - |B_j|$ vertices from set $A_j \cup \overline{B}_j$ placed to the right of cut a_j in arrangement g. $|A_j| - |B_j|$ is the number of roots in set $A_j \cap (Q - Q_3)$.

Lemma 3.7. $\forall v \in \overline{X}_j$ the child of v must be to the right of cut a_j in arrangement f.

From Lemmas 3.6 and 3.7 we can show that for each vertex $v \in Q_3$, its pair $r \in S_2$ constructed in procedure A_CUT has the following property.

Lemma 3.8. For each vertex $v \in Q_3 \cup S_1$ ($v \in Q_3 \cup S_2$), there is a corresponding vertex $r \in S_2$ ($r \in S_1$) such that

$$\frac{\sum_{w \in D(v)} C^f_k(w)}{\sum_{w \in D(v)} C^g_k(w) + C^g_k(r)} = O(\log n).$$

Proof: If v' is a child of r, then from Lemma 3.6 we have $m(v') \leq f(r) < f(v')$. Thus $r \propto v'$. Since $P(r) = 0$ and $\{r\} \cup \{D(v)\}$ is a dominating sequence with root r, similar to Lemma 3.5 we can show that

$$\frac{\sum_{w \in D(v)} C^f_k(w)}{\sum_{w \in D(v)} C^g_k(w) + C^g_k(r)} = O(\log n).$$

∎

Let $Q_4 = Q - \{Q_1 \cup Q_2 \cup Q_3\}$ and Q_4 can be partitioned into two sets Q_4' and Q_4'' with $Q_4' \subseteq S_1$ and $Q_4'' \subseteq S_2$. We will discuss set Q_4' only. Assume that cuts $a_1, \ldots, a_s$ are generated in procedure A_CUT.

Lemma 3.9. If $C^g_{\hat{a}} = \max_{i=1}^{s}\{C^g_{a_i}\}$ then

$$C^g_{\hat{a}} = \Omega\left(\frac{\sum_{q \in Q_4'} \Delta(q)}{\log^2 n}\right).$$

Proof Let $I_j = A_j \cap Q_4' = \{v^j_1, v^j_2, \ldots, v^j_{t(j)}\}$ with $\Delta(v^j_1) \geq \Delta(v^j_2) \geq \cdots \geq \Delta(v^j_{t(j)})$. This lemma can be proved by the following claim.

Claim: For each cut a_j, we have

$$\max(C^g_{a_1}, C^j_{a_2}, \ldots, C^g_{a_j}) \geq l \cdot \Delta(v^j_i)/\log n, \quad \text{for } 1 \leq l \leq t(j).$$

Let $C^g_{\hat{a}} = \max_{i=1}^{s}\{C^g_{a_i}\}$ and $Q_4' = \{v_1, v_2, \ldots, v_t\}$ with $\Delta(v_1) \geq \Delta(v_2) \geq \cdots \geq \Delta(v_t)$. From the Claim we know that

$$\Delta(v_i) \leq \frac{C^g_{\hat{a}}}{i \cdot \log n}, \quad \text{for } 1 \leq i \leq t.$$

This completes the proof. ∎

Theorem 3.1. There is a cut k_1 in arrangement g such that

$$C^g_{k_1} = \Omega\left(\frac{\sum_{v \in Q} \Delta(v)}{\log^2 n}\right).$$

Proof: From Lemmas 3.5 and 3.9 we have

$$\sum_{q \in Q_1 \cup Q_2 \cup Q_3} \Delta(q) = O(C^g_k \log n).$$

From Lemma 3.9 we know that there are two cuts $a \geq k$ and $b < k$ such that

$$\sum_{q \in Q'_4} \Delta(q) = O(C^g_a \log^2 n) \quad \text{and}$$

$$\sum_{q \in Q''_4 \Delta(q)} = O(C^g_b \log^2 n).$$

Let k_1 be such that $C^g_{k_1} = \max\{C^g_k, C^g_a, C^g_b\}$, then we have

$$C^g_{k_1} = \Omega\left(\frac{\sum_{v \in Q} \Delta(v)}{\log^2 n}\right).$$

Theorem 3.1 says that, in going from arrangement f to arrangement g, if vertices in W reduce the size of cut k by δ then there exists another cut k_1, possibly the same as k, whose size is at least $\Omega(\delta/log^2 n)$. In the next section we will concentrate on handling the effect caused by vertices in Q when we go from arrangement f to arrangement g.

IV. Lower Bound on the Size of Cut k_2 in Arrangement g

In this section we first partition set $Q \cap S_1$ into four subsets P_1, P_2, P_3 and P_4. We then show that when we go from arrangement f to arrangement g vertices in P_1, P_2, and P_3 can reduce the size of cut k by at most a factor of $O(\frac{1}{\log n})$. We next show that, when going from arrangement f to arrangement g, if vertices in P_4 reduces the size of cut k by δ, then there is another cut k' in arrangement g with size at least $\Omega(\delta/\log^2 n)$. Vertices in $Q \cap S_2$ can be handled similarly. Without loss of generality, we may assume that $P(v) > 0$, for each $v \in Q$.

Let $v_1, v_2, \in B$, $f(v_1) < f(v_2)$ and $v_1 \in S_1$, we say v_1 *covers* v_2 if

$$p < \begin{cases} \lceil \frac{1}{8 \log n} |N(v_1)| \rceil & \text{if } v_1, v_2 \in R^f, \\ \lceil \frac{1}{8 \log n} P(v_1) \rceil & \text{if } v_1 \in L^f, \end{cases} \qquad \text{where}$$

$$p = \begin{cases} |\{(u, v_1) \in E \mid u \leq f(v_2)\}| & \text{if } v_1, v_2 \in R^f \\ |\{(u, v_1) \in E \mid k \leq u \leq f(v_2)\}| & \text{if } v_1 \in L^f. \end{cases}$$

Let $v_0 \in S_1$, $v_t \in S_2$, we call $v_1, v_2, \ldots, v_t$ a *cover sequence* if it satisfies the following properties:

1) $f(v_0) \leq k < f(v_1) < f(v_2) < \cdots < f(v_t)$,

2) v_i covers $v_{i+1}, 0 \leq i \leq t-1$,

or

1) $f(v_0) < f(v_1) \leq k < f(v_2) < \cdots < f(v_t)$ and $t \geq 2$.

2) v_i covers $v_{i+1}, 0 \leq i \leq t-1$

Let $v_0, v_1, \ldots, v_t$ be a cover sequence and $v_0 \in Q \cap S_1$. There are two cases: either (1) $P(v_0) \geq 8 \log n$ or (2) $P(v_0) < 8 \log n$. Properties of a cover sequence will be discussed in the following lemmas. Lemmas 4.1 and 4.2 are for Case 1, and Lemmas 4.3 and 4.4 are for Case 2.

Lemma 4.1 If $v_0, v_1, \ldots, v_{t+1}$ is a cover sequence, and $P(v_0) \geq 8 \log n$, then $|N(v_{t+1})| \geq 0.5 \cdot P(v_0)$.

Lemma 4.2. Let $v \in S_1, r \in S_2$ and $P(v) \geq 8 \log n$. If $|N(r)| < 0.5 \cdot P(v)$, then there is a vertex x, $k < f(x) \leq f(r)$ such that for each vertex y, $f(v) \leq f(y) < f(x)$, y does not cover vertex x.

Proof: Suppose such vertex x does not exist. There is at least a vertex v_t which covers r with $f(v_t) < f(r)$. Similarly, there is also a vertex v_{t-1} which covers v_t. We can obtain a cover sequence $v, v_1, \ldots, v_t, r$ by repeatedly using this argument. From Lemma 4.1, we have $|N(r)| \geq 0.5P(v)$. A contradiction is obtained. ∎

Let $P_1 = \{\, v \in Q_3 \cap S_1 \mid P(v) < 8 \log n \,\}$ and $P_2 = \{\, v \in Q_4 \cap S_1 \mid P(v) < 8 \log n \,\}$, where Q_3 and Q_4 are the sets constructed in procedure A_CUT in Section III. Lemma 4.3 and 4.4 can be easily obtained by the properties of Q_3 and Q_4.

Lemma 4.3. For each vertex $v \in P_1$, there is a corresponding vertex $r \in S_2$, such that

$$\frac{C_k^f(v)}{C_k^g(r) + C_k^g(v)} = O(\log n).$$

Lemma 4.4. Cut a_s constructed in procedure A_CUT has the following property:

$$\frac{\sum_{v \in P_2} P(v)}{C^g_{a_s}} = O(\log n).$$

Let $\overline{X_i}, \overline{Y_i}$ are the same sets generated in Section III. We use the following procedure B_CUT to generate the sets $P_3, P_4 \subseteq (Q - P_1 - P_2)$

```
procedure B_CUT
    $P_3, P_4 = \emptyset$; $U = \emptyset$;
    for $i = 0$ to $l$ do
        Let $U = U \cup \overline{X_i} = \{x_1^i, x_2^i, \ldots, x_{s(i)}^i\}$ with $P(x_1^i) \geq P(x_2^i) \geq \cdots \geq P(x_{s(i)}^i)$.
        Let $\overline{Y}_{i+1} = \{y_1^{i+1}, y_2^{i+1}, \ldots, y_{t(i+1)}^{i+1}\}$ with $f(y_1^{i+1}) < f(y_2^{i+1}) < \cdots < f(y_{t(i+1)}^{i+1})$.
        $j = 1$;
        while ($U \neq \emptyset$) and ($j \leq t(i+1)$) do
            if $P(y_{j+1}^i) > 0$ then $P_4 = U \cup P_4$; $U = \emptyset$; exit while loop; endif;
            Choose the smallest $r$ such that $0.5 \cdot P(x_r^i) < |N(y_{j+1}^i)|$ and
            $\forall r' < r,\ 0.5 \cdot P(x_{r'}^i) > |N(y_{j+1}^i)|$
            $P_3 = P_3 \cup \{x_r^i\}$; $P_4 = P_4 \cup \{\, x_{r'}^i \mid x_{r'}^i \in U$ and $r' < r \,\}$;
            $U = U - P_3 - P_4$;
            $j = j + 1$;
        endwhile;
    endfor;
    Let $P_3 = P_3 - P_1 - P_2$; $P_4 = P_4 - P_1 - P_2$;
end; { of B_CUT }
```

Since each vertex $v \in P_3$ is paired with a vertex $r \in S_2$ in procedure B_CUT, we can otain the following lemma easily.

Lemma 4.5. For each vertex $v \in P_3$ there is a corresponding vertex $r \in S_2$ such that

$$C_k^f(v) = O(C_k^g(r)).$$

We define the *cover range function*, $\alpha: S_1 \rightarrow A \cup \{0\}$, in arrangement f as follows:

$$\alpha(v) = \min_{x \in B}\{\, f(x) \mid \forall y,\ \text{if } f(v) \leq f(y) < f(x) \text{ then } y \text{ does not cover } x \,\}$$

We construct cut b_j in the following way:

$$W_i = \bigcup_{j=1}^{i} \overline{X_j} \cap P_4 = \{x_1, x_2, \ldots, x_{t(i)}\},$$
$$\text{where } \ \alpha(x_1) \leq \alpha(x_2) \leq \cdots \leq \alpha(x_{t(i)})$$

Let b_j be the cut $(\alpha(x_j) - 1)$, where $x_j \in \bigcup W_i$. The following lemma shows the effect of vertices in the set P_4 on the cuts in arrangement g.

Lemma 4.6. Let $b_1, \ldots, b_s$ be cuts generated as above. If $C_b^g = \max_{i=1}^{s}\{C_{b_i}^g\}$, then

$$C_b^g = \Omega\left(\frac{\sum_{q \in P_4} C_k^f(q)}{\log^2 n}\right).$$

Theorem 4.1. There is a cut k_2 in arrangement g such that

$$C_{k_2}^g = \Omega\left(\frac{\sum_{v \in Q} C_k^f(v)}{\log^2 n}\right).$$

Proof: The proof of this theorem is similar to the proof of Theorem 3.1 in the previous chapter and is omitted. ■

Theorem 4.1 together with Theorem 3.1 give us the main theorem of this paper.

Theorem 4.2. Let G be a bipartite graph and f be a bipartite arrangement of G that minimizes the total edge length, then

$$D(f) = O(D^*(G)\log^2 n).$$

V. Conclusion

In this paper we establish a relationship between the density of a bipartite graph and the density of an arrangement that minimizes the total edge length. We show that if f is an arrangement that minimizes the total edge length then $D(f) = O(D^*(G)\log^2 n)$. Since f can be determined by a polynomial time algorithm [BL87], we have a $\log^2 n$-approximation algorithm for solving the density problem.

VI. References

[BL87] Bui, T. N. and S. Lee, “On the Mincut Bipartite Arrangement Problem,” Proceedings of the 1987 International Conference on Computer-Aided Design, pp. 466–469.

[PDS] Persky, G, D. N. Deutsch, and D. G. Schweikert, “LTX – A System for the Directed Automatic Design of LSI Circuits,” 13th Design Automation Conference, 1976, pp. 399–407.

[PK] Preas, B. T. and P. G. Karger, “Automatic Placement: A Review of Current Techniques,” Proceedings of the 23th Design Automation Conference, 1986, pp. 622–629.

A LINEAR TIME ALGORITHM FOR VERTEX-DELETION BIPARTIZATION ON OUTERPLANAR GRAPHS

HYEONG-AH CHOI
Department of Electrical Engineering and Computer Science
George Washington University
Washington, DC 20052

ABSTRACT

The vertex-deletion graph bipartization problem is concerned with the deletion of a set of vertices from a graph so as to make the remaining graph bipartite. The problem is known to be NP-complete even for planar graphs. In this paper we show that if the graph is outerplanar, such a vertex set of minimum cardinality can be found in linear time.

1. INTRODUCTION

Let $G(V,E)$ be a graph with vertex set V and edge set E. The degree of vertex v, denoted by $d_G(v)$, is the number of edges incident at v in G. Let $\Delta(G) = \max\{d_G(v) \mid v \in V\}$. A graph $G(V,E)$ is called *bipartite* if V can be partitioned into two disjoint subsets V_1 and V_2 such that for every edge $e \in E$, e has one end point in V_1 and the other end point in V_2. The *vertex-* (resp., *edge-*) *deletion graph bipartization* problem is the problem of deleting a set of vertices (resp., edges) from a graph so that the remaining graph becomes bipartite. A graph is called *planar* if it can be drawn in the plane with no edges crossing. A graph is called *outerplanar* if it can be drawn in the plane with no edges crossing in such a way that all vertices lie on the exterior face. A graph is called *maximal outerplanar* if it is outerplanar and no edge between nonadjacent vertices can be added without loosing outerplanarity.

Garey, Johnson and Stockmeyer [4] and Yannakakis [8] proved that the edge-deletion graph bipartization problem is NP-complete even if G is a cubic graph, *i.e.*, $d_G(v) = 3$ for all $v \in V$. Using this result, as indicated by Choi, Nakajima and Rim [3], one can easily show by constructing the *line* graph of a cubic graph that the vertex-deletion graph bipartization problem is NP-complete for a general graph with $\Delta(G) = 4$. Choi, et. al [3] showed that the NP-completeness result for the vetex-deletion graph biparzation problem still holds for a graph with $\Delta(G) = 3$. However, if the graph is planar, the edge-deletion graph bipartization problem is solvable in polynomial time due to Hadlock [5]. But the vertex-deletion graph bipartization problem was shown to be NP-complete for planar graphs with $\Delta(G) = 6$ by Yannakakis [8] and Krishnarmoorthy and Deo [7]. Recently, Choi, et. al [3] showed that the vertex-deletion graph bipartization problem still remains NP-complete for planar graphs with $\Delta(G) = 4$. They [3] also showed that if G is planar with $\Delta(G) \leq 3$, there exists a polynomial time algorithm to delete a minimum number of vertices so as to make the remaining graph bipartite.

In the following section, we consider the vertex-deletion graph bipartization problem when the graph is outerplanar.

2. A LINEAR TIME ALGORITHM

Given an outerplanar graph, our algorithm finds a subset of vertices of minimum cardinality whose removal makes the remaining graph bipartite. The algorithm is in two phases. In the first phase, the vertex-deletion graph bipartization problem is transformed to the *vertex cover* problem, where the vertex cover problem is a problem of finding a subset of vertices such that every edge in the graph has at least one end point in that subset. A vertex cover of minimum cardinality of an outerplanar graph can be found in $O(|V|)$ time due to Baker [1]. Using this result, a solution to the problem, *i.e.*, a set of vertices to be deleted in order to have the remaining graph bipartite, can be found in the second phase of the algorithm. Each phase takes $O(|V|)$ time. Thus, the overall time complexity of our algorithm is $O(|V|)$ time.

The first phase of the algorithm is based on the well-known result that a graph G is bipartite if and only if there exists no cycle of odd lengh in G. The size, in number of vertices, of the outerplanar graph is recursively reduced by one, while maintaining the resulting graph outerplanar, until it becomes a triangle. Furthermore, all the necessary information is preserved before obtaining a reduced outerplanar graph. More precisely, it is described as follows. Given an outerplanr graph $G(V,E)$, construct a maximal outerplanar graph $G_1(V,E)$. This can be done in $O(|V|)$ time as indicated in [2]. Label each edge $(u,v) \in E(G_1)$ with a number $l(u,v) \in \{0, 1, 2\}$ such that initially, $l(u,v) = 1$ if $(u,v) \in E(G)$ and $l(u,v) = 0$ otherwise. Let G_0 be a copy of G_1. Let S and P denote two sets, which are initially empty. The vertices of G_1 are scanned sequentially and those of degree 2 vertices become elements of set S. (Note that every maximal outerplanar graph has at least two vertices of degree 2 [5].) The following steps are repeated until G_1 is reduced to a triangle.

```
Step 1. Remove a vertex $v$ from $S$.
Step 2. Determine the vertices $u$ and $w$ which are incident at $v$ in $G_1$.
Step 3. If $l(u,v) \neq 0$ and $l(v,w) \neq 0$
            then if $l(u,w) = 0$
                      then if $l(u,v) + l(v,w) =$ odd
                                then $l(u,w) \leftarrow 1$
                                else $l(u,w) \leftarrow 2$
                      else if $l(u,v) + l(v,w) + l(u,w) =$ odd
                                then $l(u,w) \leftarrow 0$ and $P \leftarrow P \cup \{<u,w>\}$.
Step 4. Let $V(G_1) \leftarrow V(G_1) - \{v\}$ and $E(G_1) \leftarrow E(G_1) - \{(u,v), (v,w)\}$.
Step 5. If $d_{G_1}(u)$ (resp., $d_{G_1}(w)$) = 2, add $u$ (resp., $w$) to $S$.
```

Let $G_1(V,E)$ be a maximal outerplanar graph constructed from a given outerplanar $G(V,E)$. To explain the idea behind the labeling process, let u, v and w be vertices as defined in *Steps* 1 and 2. Suppose $(u,v) \notin E(G)$ or $(v,w) \notin E(G)$. This case is represented by $l(u,v) = 0$ or $l(v,w) = 0$, respectively, and a solution to $G - \{v\}$ is also a solution to G. Thus *Step* 3 is skipped in this case. Suppose $(u,v) \in E(G)$ and $(v,w) \in E(G)$, represented by $l(u,v) = 1$ and $l(v,w) = 1$. If $(u,w) \in E(G)$, then u, v and w form a cycle of length 3. Clearly, at least one of these three vertices must be in the final solution. Having u or w in the final solution breaks the odd cycle $u-v-w-u$ and all other cycles, if any, containing edge (u,w). Thus, by putting the pair of vertices $<u,w>$ in P and changing the label $l(u,w)$ from 1 to 0, a solution to $G - \{v\}$ plus either vertex u or vertex w becomes a solution to G. Suppose $(u,v) \in E(G)$ and $(v,w) \in E(G)$, but $(u,w) \notin E(G)$. Then any cycle in G containing v must contain edges (u,v) and (v,w). Thus, by replacing $l(u,w)$ from 0 to 2, edge (u,w) in $G - \{v\}$ represent a path $u-v-w$ in G of length 2. Again, a solution to $G - \{v\}$ becomes a solution to G. From the second iteration of the above steps, $l(e) = 1$ and $l(e) = 2$ represent the lengths of paths in the previous graph G_1 of odd parity and even parity, respectively, and the idea of labeling process can be similarly explained. Every time *Steps* 1-5 are applied, the number of vertices in G_1 is reduced by one and G_1 remains a maximal outerplanar graph. Upon completion, G_1 is a triangle. Let i, j, k be the vertices of the triangle G_1. We next construct a graph $G_2(V,E)$ such that $V(G_2) = V(G) = V(G_0)$ and $E(G_2) = \{ (u,v) \mid <u,v> \in P \}$. (Note that $E(G_2) \subseteq E(G_0)$, which implies that G_2 is an outerplanar graph.) If $l(i,j) + l(j,k) + l(k,i) =$ odd, and each of these numbers is nonzero, then we do the following: set $TAG = 1$ and for each $(u,v) \in \{(i,j), (j,k), (k,i)\}$, construct a graph $G_2 \cup \{(u,v)\}$ from G_2 by adding edge (u,v). Otherwise, set $TAG = 0$.

We now describe the second phase of the algorithm. If $TAG = 0$, then let V^* be a minimum vertex cover for G_2. If $TAG = 1$, then find minimum vertex covers for $G \cup \{(i,j)\}$, $G \cup \{(j,k)\}$ and $G \cup \{(k,i)\}$, and let V^* be the smallest one among the three vertex covers. If there exists a tie, then arbitrarily choose one. Return V^* as a final solution.

REFERENCES

[1] B. Baker, "Approximation Algorithms for NP-complete Problems on Planar Graphs," *24th Annual Symposium on Foundations of Computer Science,* November, 1983, pp. 265-273.

[2] T. Beyer, W. Jones and S. L. Mitchell, "Linear Algorithms for Isomorphism of Maximal Outerplanar Graphs," *J. ACM,* 26, 1979, pp. 603-610.

[3] H.-A. Choi, K. Nakajima and C. Rim, *SIAM J. Disc. Math,* Vol. 2, No. 1, pp.38-47, 1989.

[4] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some Simplified NP-complete graph problems," *Theoretical Computer Science,* Vol. 1, pp.237-267, 1976.

[5] F. Hadlock, "Finding a Maximum Cut of a Planar Graph in Polynomial Time," *SIAM J. Comput.,* 4 (1975), pp. 221-225.

[6] F. Harary, *Graph Theory,* Addison-Wesley, Reading (1969).

[7] M. S. Krishnarmoorthy and N. Deo, "Node-deletion NP-complete problems," *SIAM J. Comput., 8 (1979), pp.619-623.*

[8] M. Yannakakis, "Node- and Edge-deletion NP-complete Problems," *Proc. 10th Annual ACM Symposium on Theory of Computing, San Diego, CA, October, 1978, pp.253-264.*

FILTRATIONS FOR LINEAR SYSTEMS

BOSTWICK F. WYMAN
Mathematics Department
The Ohio State University
Columbus, Ohio 43210

MICHAEL K. SAIN
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

It has been known for a long time that the controllability indices of a strictly proper transfer function coincide with the "column degrees" or the "dynamical indices" associated with certain matrix factorizations. This paper gives a new structural interpretation of this fact. Pole modules are given structures of "filtered vector space", corresponding to controllability, observability, and system structure at infinity. It is shown that the usual pole module of a strictly proper transfer function is isomorphic to a pole module at infinity arising from matrix factorization theory, and that this isomorphism respects the filtration structure. Numerical corollaries give the classical results on indices. The present paper should be considered a preliminary report, since only the classical strictly proper case is considered. Conjecturally a full-fledged theory of filtrations on global pole and zero spaces should become available.

1. INTRODUCTION

Suppose given a strictly proper transfer function $G(z) : U(z) \to Y(z)$ with a matrix factorization $G(z) = N(z)D(z)^{-1}$ in which $N(z)$ and $D(z)$ are coprime polynomial matrices, and which has the property that the matrix $\begin{pmatrix} D \\ N \end{pmatrix}$ is column proper. Then the "dynamical indices" or "column degrees" of $\begin{pmatrix} D \\ N \end{pmatrix}$ are given as the maximum degrees of the polynomials in each column. In fact, since $G(z)$ is strictly proper, the dynamical indices of $D(z)$ are exactly the same as the ones for $\begin{pmatrix} D \\ N \end{pmatrix}$ and they are exactly the controllability indices of G. See [Fo; Ka, Chap. 6] for these results. The goal of this preliminary and expository note is to give a new structural interpretation and to indicate the proof of this result and its analog for observability indices. Roughly speaking, the ideas go like this: If we consider $D(z)$ as a polynomial (improper!) transfer function $D(z) : U(z) \to U(z)$, then we can compute the pole module at infinity, $X_\infty(D)$. This pole module is a module over the ring $\mathcal{O}_\infty$ of proper rational functions, or equivalently a finite dimensional vector space equipped with a nilpotent dynamics map defined by the action of $1/z$ [CP1-2]. Now $\mathcal{O}_\infty$ is a principal ideal domain, so the pole module $X_\infty(D)$ is determined (up to isomorphism) by its invariant factors, all of which are of the form $(1/z)^e$ for various integer exponents e. The exponents of these invariant factors can be viewed as the sizes of the Jordan blocks of the nilpotent $1/z$ action, and, most importantly for us, they coincide with the dynamical indices of $D(z)$. This fact is "folklore", and a proof can be found in [WSCP2, Sec. 6]. The identification of these numbers with controllability indices will then follow from an isomorphism of the ordinary pole module $X(G)$ (the minimal realization state space for $G(z)$) with the pole module $X_\infty(D)$. Since these two spaces cannot be module-isomorphic (since they are modules over different rings), and since isomorphism as vector spaces is not sufficient for our needs, we need to seek some kind of intermediate structure on all these spaces and isomorphisms which preserve this structure. Fundamentally, we are seeking a relation between the ordinary dynamics structure on the state space $X(G)$ and the nilpotent dynamics structure on $X_\infty(D)$. The relevant idea here is the notion of a "filtered vector space", which will be discussed in the next section.

2. FILTRATIONS

Suppose V is any finite dimensional vector space. Then a *filtration* of V is an increasing sequence of subspaces

$$0 = V_0 \subset V_1 \subset \ldots \subset V_k = V.$$

If all the subspace inclusions are proper, we say that we have a *strict filtration.* Associated to a filtration with k nonzero spaces, there are k positive integers $v_1, v_2, \ldots, v_k$ defined by

$$v_i = \dim V_i - \dim V_{i-1}, \text{ or}$$
$$\dim V_i = v_1 + v_2 + \cdots + v_i.$$

Since $\dim V_k = \dim V = n$, then the integers v_i form a **partition of** n. (That is, $v_1 + v_2 + \cdots + v_k = n$). The **Ferrers diagram** of a partition is an arrangement of dots such that the i^{th} row has v_i dots, ignoring any cases where $v_i = 0$. (See [L, Ri] for more discussion and examples.) The **conjugate partition of** n is the sequence of integers giving the length of the columns of the Ferrers diagram.

Suppose given a system (A,B,C) on a state space X, assuming that (A,B) is controllable and that (A,C) is observable. Two partitions on X are familiar objects in control theory.

The **controllability filtration** of (A,B) is just the sequence of column spaces

$$[B] \subset [B|AB] \subset \ldots \subset [B|AB|\ldots|A^kB] \subset \ldots$$

We stop when we get to X, which happens by the controllability assumption. Standard lemmas show that the filtration is strictly increasing. The **controllability indices** for (A,B) are defined by the *conjugate partition* to the partition derived from the controllability filtration. (See also the discussion of "crates" in [Ka, p. 424 ff.].)

The **observability filtration** of (A,C) is the sequence of nullspaces

$$\ldots \subset \ker \begin{bmatrix} C \\ \vdots \\ CA^k \end{bmatrix} \subset \ldots \subset \ker \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} \subset \ker \begin{bmatrix} C \\ CA \end{bmatrix} \subset \ker[C].$$

The subspace inclusions in this filtration are strict, and for sufficiently large k, the kernel is zero, by our observability hypothesis. The **observability indices** for (A,B) are defined by the *conjugate partition* to the partition derived from the observability filtration.

The third partition we will be concerned with is the **kernel filtration of a nilpotent linear transformation**. Let M be a finite-dimensional $\mathcal{O}_\infty$ - module, viewed as a vector space with a nilpotent map $J : M \to M$. Let k be the smallest integer such that $J^k = 0$. Define a filtration by $M_i = \ker(J^i)$, so $M_k = M$. The corresponding partition of $\dim M$ has the following geometric meaning: v_1 is the number of eigenvectors of J (one eigenvector for each Jordan block, as usual), v_2 is the number of generalized eigenvectors of length 2, and so on. It is not too hard to see that the conjugate partition gives the number of Jordan blocks of various dimension: for example, a block of size 3 gives one eigenvector, one generalized eigenvector of length 2, and one of length 3. (The "length" of a vector is the number of iterations of J it takes to send a vector to zero.) In other words the conjugate partition of the kernel filtration gives the (exponents of the) invariant factors of the nilpotent matrix J.

3. FILTRATIONS AND CONTROLLABILITY INDICES

The new structural result we have in mind is given by the following theorem.

Controllability Filtration Theorem: Suppose $G(z) = N(z)D^{-1}(z)$ is a strictly proper transfer function with $N(z)$ and $D(z)$ right coprime and $\begin{pmatrix} D \\ N \end{pmatrix}$ column proper. Then there is an isomorphism of filtered vector spaces

$$\delta : X_\infty(D) \cong X(G),$$

where $X_\infty(D)$ is given the kernel filtration and $X(G)$ is given the controllability filtration.

In this context an "isomorphism of filtered vector spaces" is an isomorphism of ordinary vector spaces (i.e., a non-singular linear transformation) which preserves the filtrations. Thus, if

$$0 = K_0 \subset K_1 \subset \ldots \subset K_t = X_\infty(D)$$

is the kernel filtration on $X_\infty(D)$, and the controllability filtration on $X(G)$ is

$$0 = C_0 \subset C_1 \subset \ldots \subset C_s = X(G),$$

then $s = t$ and the mapping δ maps K_i onto C_i for each i. Suppose the mapping δ has been constructed and the theorem has been proved. Then the partitions corresponding to the two filtrations coincide, so that the conjugate partitions also coincide. This is exactly the statement that the invariant factors of $X_\infty(D)$ coincide with the controllability indices of $G(z)$. Finally, we need to recall that the invariant factors of $X_\infty(D)$ are given by the column degrees of the matrix D or $\begin{pmatrix} D \\ N \end{pmatrix}$, as mentioned above.

We conclude this section by giving a brief sketch of the proof. We proceed by giving an abstract algebraic description of the pole modules involved and of their filtrations. We define the mapping δ and verify that it maps the kernel filtration *into* the controllability filtration. The fact that δ is one-to-one and onto is best understood in the general context of [WSCP1-2] and will be discussed in Section 5.

The pole module at infinity corresponding to the polynomial matrix D can be described informally as

$$X_\infty(D) = \frac{\text{strictly proper input vectors}}{\text{vectors whose D-image is also strictly proper}}.$$

The module of strictly proper input vectors is denoted $z^{-1}\Omega_\infty(U)$, so the formal description of this pole module is

$$X_\infty(D) = \frac{z^{-1}\Omega_\infty(U)}{z^{-1}\Omega_\infty(U) \cap D^{-1}(z^{-1}\Omega_\infty(U))}.$$

The informal description of the ordinary pole module is

$$X(G) = \frac{\text{polynomial input vectors}}{\text{vectors whose G-image is also polynomial}}$$

or $X(G) = \frac{\Omega U}{\Omega U \cap G^{-1}(\Omega Y)}$, where $\Omega U = U[z]$ is just the free module of polynomial vectors with coefficients in U. With these descriptions of the modules, the map δ is induced from the map

$$u(z) \to \pi_+ D(z)u(z)$$

where $u(z)$ is a strictly proper representative defining a class in $X_\infty(D)$, and $\pi_+(D(z)u(z))$ is the polynomial part of its image in $U(z)$. If $u(z)$ is zero in $X_\infty(D)$, then $D(z)u(z)$ is strictly proper, π_+ gives 0, and the map δ is well defined.

Now let $J : X_\infty(D) \to X_\infty(D)$ be the nilpotent map induced by $1/z$. We have

$$\begin{aligned} \ker J^i &= \{u(z) : D(z)z^{-i}u(z) \in z^{-1}\Omega_\infty(U)\} \\ &= \{u(z) : D(z)u(z) \in z^{i-1}\Omega_\infty(U)\} \\ &= \{u(z) : \pi_+ D(z)u(z) \in U + Uz + \cdots + Uz^{i-1}\}. \end{aligned}$$

Now $U + Uz + \cdots + Uz^{i-1}$ is a subspace of ΩU whose image in $X(G)$ is exactly the column space of $[B|AB|\ldots|A^{i-1}B]$ in the usual notation verifying that the kernel filtration maps exactly over to the controllability filtration. This gives the main idea of the proof, except for the verification that δ is bijective.

The numerical implications of the theorem, which are well-known and motivated the new structural results, can be summarized in the following corollary. We omit the proof, which has mostly been sketched already.

Corollary (Controllability Case): Assume given $G(z) = N(z)D^{-1}(z)$ as in the Controllability Filtration Theorem. Then the following sets of integers coincide.

(a) The column indices of $D(z)$.
(b) The column indices of $\begin{pmatrix} D \\ N \end{pmatrix}$.
(c) The invariant factor exponents of $X_\infty(D)$.
(d) The invariant factor exponents of $X_\infty(\begin{pmatrix} D \\ N \end{pmatrix})$.
(e) The controllability indices of $G(z)$.

4. FILTRATIONS AND OBSERVABILITY INDICES

We state here a sort of "dual" to the Controllability Filtration Theorem.

Observability Filtration Theorem: Suppose $G(z) = D^{-1}(z)N(z)$ is a strictly proper transfer function with $N(z)$ and $D(z)$ left coprime and $(D\ N)$ row proper. Then there is an isomorphism of filtered vector spaces

$$\delta : X_\infty(D) \cong X(G),$$

where $X_\infty(D)$ is given the kernel filtration and $X(G)$ is given the observability filtration.

As before, there is a numerical corollary which motivated the structural result.

Corollary (Observability Case): Assume given $G(z) = D^{-1}(z)N(z)$ as in the Observability Filtration Theorem. Then the following sets of integers coincide.

(a) The row indices of $D(z)$.
(b) The row indices of $(D\ N)$.
(c) The invariant factor exponents of $X_\infty(D)$.
(d) The invariant factor exponents of $X_\infty(D\ N)$.
(e) The observability indices of $G(z)$.

5. PROOFS OF ISOMORPHISMS VIA THE BASIC POLE-ZERO SEQUENCE

In this section we assume that the reader is familiar with the recent results in [WSCP1-2]. In particular we use without further explanation the ideas of Global Pole Spaces, Global Zero Spaces, and Wedderburn-Forney spaces, and the Basic Pole-Zero Exact Sequence (Theorem 5.1 of [WSCP2]).

Suppose $G(z) = N(z)D^{-1}(z)$ is given as in the Controllability Filtration Theorem. Let $H : U(z) \to U(z) \oplus Y(z)$ be given by $H(z) = \begin{pmatrix} D \\ N \end{pmatrix}$. Then since $H(z)$ is injective the Basic Pole-Zero Exact Sequence gives

$$0 \to \mathcal{Z}(H) \to \mathcal{X}(H) \to \mathcal{W}(\text{ image } H) \to 0,$$

where $\mathcal{Z}()$ denotes global zeros, $\mathcal{X}()$ denotes global poles, and $\mathcal{W}()$ denotes the Wedderburn-Forney construction. We have assumed that $N(z)$ and $D(z)$ are right coprime, and that H is column proper. In other words, the columns of H give a minimal polynomial basis of its column space, so that $\mathcal{Z}(H) = 0$ ([WSCP2, Th. 6.3]). Since H is a polynomial matrix $\mathcal{X}(H) = X_\infty(H)$, which in turn is just $X_\infty(D)$ since $G(z)$ is strictly proper. That is, the exact sequence supplies an isomorphism of vector spaces

$$X_\infty(D) \cong \mathcal{W}(\text{ image } H).$$

Next, we apply these same ideas to the closely related matrix $\begin{pmatrix} I \\ G \end{pmatrix}$. The exact sequence reads

$$0 \to \mathcal{Z}(\begin{pmatrix} I \\ G \end{pmatrix}) \to \mathcal{X}(\begin{pmatrix} I \\ G \end{pmatrix}) \to \mathcal{W}(\text{ image } \begin{pmatrix} I \\ G \end{pmatrix}) \to 0.$$

The presence of the identity mapping I insures that there are no global zeros at all, and the space $\mathcal{X}(\begin{pmatrix} I \\ G \end{pmatrix})$ is isomorphic to $\mathcal{X}(G)$, which is just $X(G)$ (the finite pole module) since $G(z)$ is strictly proper. That is, we have another vector space isomorphism

$$X(G) \cong \mathcal{W}(\text{ image } \begin{pmatrix} I \\ G \end{pmatrix}).$$

On the other hand the column space of H over $k(z)$ is exactly the same as the column space of $\begin{pmatrix} I \\ G \end{pmatrix}$, since $D(z)$ is invertible. Since the Wedderburn construction depends only on the column space, $\mathcal{W}(\text{ image } H)$ is isomorphic to $\mathcal{W}(\text{ image } \begin{pmatrix} I \\ G \end{pmatrix})$. Pasting together the two displayed isomorphisms then gives

$$X_\infty(D) \cong X(G),$$

and chasing through the explicitly defined mappings from [WSCP2] shows that this isomorphism is exactly the map δ in the Controllability Filtration Theorem. In fact, the definition of δ was inspired directly by the Basic Exact Sequence. The new idea in this paper is exactly the introduction of filtrations and the verification that δ is an isomorphism, not just of vector spaces, but of filtered vector spaces.

The proof of the Observability Filtration Theorem follows in much the same way from the two exact sequences associated to the mappings $[I|G(z)]$ and $[D(z)|N(z)]$ where $G(z) = D(z)^{-1}N(z)$ is a left coprime, row proper matrix factorization. These two mappings are surjective rather than injective, so it is necessary to apply the Wedderburn-Forney construction to their kernels. Details are omitted here.

6. CONCLUSIONS AND WORK IN PROGRESS

We have tried to indicate in this brief introductory paper the role of filtered vector spaces in linear system theory, and in particular their relation to various famous sets of integers: controllability indices, observability indices, and column degrees. We have restricted our attention to strictly proper transfer functions, and our contributions are in the formalization process and in some new proofs. In fact, many of these ideas, and certainly the numerical results, have been known for about twenty years.

The importance of the Basic Pole-Zero Exact Sequence really depends on the fact that it applies to improper transfer functions and the corresponding singular systems just as to the classical cases discussed here. The goal of our ongoing investigation is to establish filtrations on the three kinds of spaces, namely global pole spaces, global zero spaces, and the Wedderburn-Forney spaces. In fact, two new filtrations have been established for poles and Wedderburn spaces, corresponding to the control and observation points of view, and supply global controllability and observability indices. A complete treatment of these questions will supply a deep generalization of the relations between column degrees and controllability and observability indices established in the present paper. See [WCP] for a first attempt.

REFERENCES

[CP1] G. Conte, A. Perdon, "Generalized State Space Realizations of Non-Proper Transfer Functions", *Systems and Control Letters*, 1, 1982.

[CP2] G. Conte, A. Perdon, "Infinite Zero Module and Infinite Pole Module", *VII Int. Conf. Analysis and Opt. of Systems: Nice*, Lec. Notes Control and Inf. Science, Springer 62, pp. 302-315, 1984.

[Fo] G. D. Forney, Jr., "Minimal Bases of Rational Vector Spaces with Applications to Multivariable Linear Systems", *SIAM J. Control*, 13, 493-520, 1975.

[Ka] T. Kailath, *Linear Systems*, Prentice Hall: Englewood Cliffs, 1980.

[L] W. Ledermann, *Introduction to Group Characters*, Second Edition, Cambridge U.P., 1987. (Section 4.10).

[R] H. Rosenbrock, *State Space and Multivariable Theory*, Wiley: New York, 1970.

[Ri] J. Riordan, *An Introduction to Combinatorial Analysis*, Wiley, 1958. Reprinted, Princeton U.P., 1980. (Chapter 6).

[WCP] B. Wyman, G. Conte, A.-M. Perdon, "Algebraic and Geometric Indices for Singular Systems", 1989 IEEE Conference on Decision and Control, to appear.

[WSCP1] B. Wyman, M. Sain, G. Conte, A.-M. Perdon, "Rational Matrices: Counting the Poles and Zeros", 1988 IEEE Conference on Decision and Control, Austin, TX.

[WSCP2] B. Wyman, M. Sain, G. Conte, A.-M. Perdon, "On the Zeros and Poles of a Transfer Function", to appear, Special Issue on Systems and Control, *Linear Algebra and Applications*, 1989.

DISCRETE-TIME p-POWERS AND STABILITY *

IRWIN W. SANDBERG
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712

ABSTRACT

Iterated-integral operators called p-powers arise in the theory of representations of nonlinear systems. In a recent paper, necessary and sufficient conditions are given for such operators to be stable in a standard sense, and it is shown that a well-known sufficient condition is not necessary for all $p \geq 2$. Here corresponding results are given for discrete-time cases.

I. INTRODUCTION

In [1] several results are given concerning the stability of p-powers and p-linear maps. Attention is focused on p-powers H defined by

$$(Hx)(t) = \int_0^t \cdots \int_0^t \ell(t, \tau_1, \ldots, \tau_p) x(\tau_1) \ldots x(\tau_p) d\tau_1 \ldots d\tau_p, \quad t \geq 0 \tag{1}$$

for x a general element of L_∞, with weak conditions (see [1]) on ℓ. The paper's main result is a criterion for H to be stable in the sense that $H(L_\infty) \subset L_\infty$, and it is shown that a well-known sufficient condition is not necessary for all $p \geq 2$. Results of this kind are of interest in the theory of systems because, for example, operators of the type (1) arise in connection with Volterra expansions for input-output maps.

This paper reports on corresponding results for the discrete-time case in which, instead of (1), H is given by

$$(Hx)(t) = \sum_{\tau_p=1}^{t} \cdots \sum_{\tau_1=0}^{t} \ell(t, \tau_1, \ldots, \tau_p) x(\tau_1) \ldots x(\tau_p), \quad t \in N \tag{2}$$

where N denotes $\{0, 1, \ldots\}$, and x and the kernel ℓ are real-valued functions defined on N and $\{(t, \tau_1, \ldots, \tau_p) \in N^{(p+1)} \; : \; t \geq \tau_j \geq 0, \text{ all } j\}$, respectively. The following is a brief summary of our results, which are given in Sections II and III.

We show that H in (2) takes ℓ_∞ into itself [†] if and only if

$$\sup_t \ \max_{J(t)} \Big| \sum_{J_p(t)} \cdots \sum_{J_1(t)} \mathrm{Sym}[\ell(t, \tau_1, \ldots, \tau_p)] \Big| < \infty, \tag{3}$$

---

[*] A summary of the results described in this paper is contained in the proceedings of the 1989 International Conference on Circuits and Systems, Nanjing, China, July 1989. However, although the proceedings is available, the conference was not held because of the political turmoil in China in June of 1989.

[†] ℓ_∞ denotes $\ell_\infty(N, \mathbb{R})$ the linear space of bounded-real valued functions on N, with the usual sup norm, and ℓ_1, meaning $\ell_1(N, \mathbb{R})$, is also defined in the usual way.

in which $J(t)$ is a general p-vector $(J_1(t), \ldots, J_p(t))$ of nonempty subsets of $\{0, 1, \ldots, t\}$ and "Sym" denotes the operation of taking the symmetric part (in the standard sense described in Section II).

For H time invariant in the sense that there is a map $k : N^p \to I\!R$ such that $k(t - \tau_1, \ldots, t - \tau_p) = \ell(t, \tau_1, \ldots, \tau_p)$, (3) can be expressed as

$$\sup_J \Big| \sum_{J_p} \cdots \sum_{J_1} \mathrm{Sym}[k(\tau_1, \ldots, \tau_p)] \Big| < \infty \tag{4}$$

where $J = (J_1, \ldots, J_p)$ is a general p-vector of nonempty finite subsets of N.

It is clear that a sufficient condition under which (4) is met is

$$\sum |\mathrm{Sym}[k(\tau_1, \ldots, \tau_p)]| < \infty, \tag{5}$$

where the sum is over $(\tau_1, \ldots, \tau_p) \in N^p$. We show that this simple condition is necessary only for $p = 1$. A simple corollary of the criterion (4) is that if k is *nonnegative*, then H is stable if and only if (5) holds.

Conditions (3) and (4) can be expressed in other forms. For example, (3) is equivalent to

$$\sup_t \sup_{J(t)} \sum_{\tau_p=0}^{\infty} \Big| \sum_{J(p-1)(t)} \cdots \sum_{J_1(t)} \mathrm{Sym}[\ell(t, \tau_1, \ldots, \tau_p)] \Big| < \infty.$$

Let $p \geq 2$, and consider the family $\{\mathrm{Sym}[k(\tau_1, \ldots, \tau_{(p-1)}, \cdot\,)] \; : (\tau_1, \ldots, \tau_{(p-1)}) \in N^{(p-1)}\}$ of functions defined on N. We show that sufficient conditions for (4) to hold are that each member of this family belongs to ℓ_1, and the family is *summable* in ℓ_1 in the sense of unordered summation. We prove that these conditions are necessary for $p = 2$.

Earlier results that bear on the $p = 2$ case involving summability (in a different but related sense) are given in [2]. The sufficiency of summability is essentially shown, but the question of necessity is not addressed. Nor is the lack of necessity of (5) (because in [2] the condition of symmetry of kernels is not imposed). An early result related to the criterion (4) is given in [3, p. 334]. For general background material, see [4]–[9].

The results given in Section II concern in addition p-linear maps K of the form

$$K(x_1, \ldots, x_p)(t) = \sum_{\tau_p=0}^{t} \cdots \sum_{\tau_1=0}^{t} \ell(t, \tau_1, \ldots, \tau_p) x_1(\tau_1) \ldots x_p(\tau_p), \quad t \in N$$

for $(x_1, \ldots, x_p) \in {\ell_\infty}^p$. These maps are of importance in connection with the representation of certain Fréchet derivatives. Also, notice that K generates H in the sense that $(Hx)(t) = K(x, \ldots, x)(t)$.

II. STABILITY OF DISCRETE-TIME p-LINEAR MAPS AND p-POWERS

2.1 *Preliminaries*

Throughout Section II, p is (as in Section I) a positive integer; $X, X_1, \ldots, X_p$ are Banach spaces, and Y is a normed linear space. The spaces may be real or complex, except that $X_1, \ldots, X_p$ are either all real or all complex. We use the same symbol $\|\cdot\|$ to denote the norm in each of the spaces, and P stands for the Banach space $X_1 \times \cdots \times X_p$ with norm $\|\cdot\|$ given by $\|x\| = \max_j \|x_j\|$ for $x = (x_1, \ldots, x_p) \in P$.

A map g from P into Y is said to be *p-linear* if $g(x_1, \ldots, x_p)$ is linear (i.e., additive and homogeneous) separately in each variable; it is well known that a p-linear g is *continuous* if and only if g is bounded in the sense that there is a positive constant c such that $\|g(x_1, \ldots, x_p)\| \leq c\|x_1\| \ldots \|x_p\|$ for all $x \in P$. Finally, a p-linear g on $P = X^p = X \times \ldots \times X$ is *symmetric* if $g(x_1, \ldots, x_p) = g(x_{\phi(1)}, \ldots, x_{\phi(p)})$ for any $x \in P$ and every permutation $\phi(1), \ldots, \phi(p)$ of $1, \cdots, p$. *

Let A be a nonempty set. In Section 2.2 we use the following extension of the Banach-Steinhaus principle of uniform boundedness proved in [3].

Theorem 1. Let $\{g_a : a \in A\}$ be a family of continuous p-linear maps from P into Y, and let σ be defined by

$$\sigma = \sup_{a \in A} \sup \{\|g_a(x_1, \ldots, x_p)\| : \|x_1\| \leq 1, \ldots, \|x_p\| \leq 1\}. \tag{6}$$

Then

$$\{\|g_a(x_1, \ldots, x_p)\| : a \in A\}$$

is bounded for each $x \in P$ if and only if $\sigma < \infty$.

Given any p-linear map f of $P = X^p$ into Y, we let $\mathrm{Sym}(f)$ denote the symmetric p-linear map on P defined by

$$\mathrm{Sym}(f)(x_1, \ldots, x_p) = \frac{1}{p!} \sum_{\phi} f(x_{\phi(1)}, \ldots, x_{\phi(p)}),$$

where $\sum_\phi$ stands for the sum over all permutation maps ϕ from $\{1, 2, \ldots, p\}$ onto itself. A second result we shall use is Theorem 2, below.

Theorem 2. Suppose that $\{h_a : a \in A\}$ is a family of maps from X into Y such that $h_a(x) = f_a(x, \ldots, x)$ for $x \in X$ and $a \in A$, where each f_a is a continuous p-linear map of X^p into Y. Let $g_a \triangleq \mathrm{Sym}(f_a)$ for each a, and let σ be given by (6). Then a necessary and sufficient condition for

$$\{\|h_a(x)\| : a \in A\}$$

---

*We define p-linearity and symmetry in the same way for X, $X_1, \ldots, X_p$, Y replaced with (not necessarily normed) linear spaces Z, $Z_1, \ldots, Z_p$, W, respectively. We take this opportunity to correct an oversight in [3] (which has no bearing on any of the results): In the definition of a symmetric map g, P should be X^p, and similarly with regard to the definition of $\mathrm{Sym}(f)$.

to be bounded for each $x \in X$ is that $\sigma < \infty$.

Theorem 2 is a variation of a corollary to Theorem 1 proved in [3]. (It is obtained from the corresponding result in [3] by just observing that $f_a(x, \ldots, x) = \mathrm{Sym}(f_a)(x, \ldots, x)$.)

2.2 Criteria for Stability

Throughout the remainder of the paper H, ℓ, K, k, ℓ_∞, ℓ_1, N, $J(t)$, and J are as described in Section I. By K *time-invariant* we mean, as in connection with (3), that there is a map $k : N^p \rightarrow I\!R$ such that $k(t - \tau_1, \ldots, t - \tau_p) = \ell(t, \tau_1, \ldots, \tau_p)$

Our key results are the following two theorems which are simple analogs of the corresponding results in [1]. In the case of Theorem 4 below, $\mathrm{Sym}[\ell(t, \tau_1, \ldots, \tau_p)]$ and $\mathrm{Sym}[k(\tau_1, \ldots, \tau_p)]$ stand for

$$\frac{1}{p!} \sum_{\phi} \ell(t, \tau_{\phi(1)}, \ldots, \tau_{\phi(p)})$$

and

$$\frac{1}{p!} \sum_{\phi} k(\tau_{\phi(1)}, \ldots, \tau_{\phi(p)}),$$

respectively, where in each case the sum is over all permutation maps ϕ on $\{1, 2, \ldots, p\}$.

Theorem 3. We have $K(\ell_\infty{}^p) \subset \ell_\infty$ if and only if

$$\sup_t \max\nolimits_{J(t)} \Big| \sum_{J_p(t)} \cdots \sum_{J_1(t)} \ell(t, \tau_1, \ldots, \tau_p) \Big| \; < \; \infty, \tag{7}$$

and for K time invariant (7) can be replaced with

$$\sup_J \Big| \sum_{J_p} \cdots \sum_{J_1} k(\tau_1, \ldots, \tau_p) \Big| \; < \; \infty. \tag{8}$$

Theorem 4. A necessary and sufficient condition for H to take ℓ_∞ into itself is that (3) holds, and for K time invariant, (3) can be replaced with (4).

Proof of Theorem 3

All proofs are omitted in this version of the paper.

2.2.1 Other Forms of the Conditions

Conditions (7) and (8) can be expressed in other forms. For example, (7) and (8), respectively, are equivalent to

$$\sup_t \sup_{J(t)} \sum_{\tau_p = 0}^{\infty} \Big| \sum_{J_{(p-1)}(t)} \cdots \sum_{J_1(t)} \ell(t, \tau_1, \ldots, \tau_p) \Big| < \infty \,, \tag{9}$$

and

$$\sup_{J} \sum_{\tau_p=0}^{\infty} \mid \sum_{J_{(p-1)}} \cdots \sum_{J_1} k(\tau_1, \ldots, \tau_p)| < \infty \, ^{*} . \tag{10}$$

By comparing (10) with condition 2.14 of [10, p.11], it follows that H with $p = 2$ takes ℓ_∞ into itself if and only if the infinite matrix A defined by $A_{\tau_1,\tau_2} = \mathrm{Sym}[k(\tau_1, \tau_2)]$ for $(\tau_1, \tau_2) \in N^2$ maps ℓ_∞ into ℓ_1 (where of course now the elements of ℓ_∞ and ℓ_1 are viewed as column vectors).

2.2.2 Summability and Stability

In this section we give sufficient conditions, involving the concept of unordered summation, under which (8) holds. For $p = 2$ we give a *complete characterization* of (8) (and therefore of the stability of H for the time-invariant case) in terms of this concept. Our result is the following:

Theorem 5. Sufficient conditions for (8) to hold, with $p \geq 2$, are

(i) each member of the family $\{k(\tau_1, \ldots, \tau_{(p-1)}, \cdot) : (\tau_1, \ldots, \tau_{(p-1)}) \in N^{(p-1)}\}$ of functions defined on N belongs to ℓ_1, and

(ii) the family is summable in ℓ_1 (in the sense of the standard definition given in the appendix).

Moreover, (i) is necessary, and for $p = 2$, so is (ii).

Proof.
Theorem 7 (in the Appendix) plays a role in the omitted proof.

Comment
Referring to the sufficiency of (i) and (ii), and the necessity of (i), a similar result can be given concerning (7) and uniform summability with respect to the parameter t.

III. THE LACK OF NECESSITY OF (5)

The following theorem shows that for every $p \geq 2$, (5) is merely a *sufficient* condition for (4) to be met.

Theorem 6. For each integer $p \geq 2$, there exists a symmetric[†] $k : N^p \to I\!R$ such that

$$\sup_{J} | \sum_{J_p} \cdots \sum_{J_1} k(\tau_1, \ldots, \tau_p)| < \infty$$

---

[*] For $p = 1$, the summation signs $\sum_{J_{(p-1)}(t)} \cdots \sum_{J_1(t)}$ and $\sum_{J(p-1)} \cdots \sum_{J_1}$ in (9) and (10) are to be ignored.

[†] Of course k *symmetric* means that $k(\tau_1, \ldots, \tau_p) = \mathrm{Sym}[k(\tau_1, \ldots, \tau_p)]$ for all $(\tau_1, \ldots, \tau_p)$.

and

$$\sum_{\tau \in N^p} |k(\tau_1, \ldots, \tau_p)| = \infty ,$$

where $\tau = (\tau_1, \ldots, \tau_p)$.

Proof of Theorem 6

Theorem 7 (in the Appendix) and results in [11] and [12] are used in the omitted proof.

APPENDIX

Unordered Summation in ℓ_r

For $r \in [1, \infty)$, let ℓ_r denote the familiar normed linear space of real-valued functions h defined on N such that $\sum_{m=0}^{\infty} |h(m)|^r < \infty$, with

$$\|h\| = \left(\sum_{m=0}^{\infty} |h(m)|^r \right)^{\frac{1}{r}} .$$

Let $p \geq 2$, and let f map $N^{(p-1)}$ into ℓ_r. Let F denote the family of all finite subsets of $N^{(p-1)}$, and let $I(A)$ be defined by

$$I(A) = \sum_{a \in A} f(a)$$

for each $A \in F$, with $I(A) = 0$ if A is the empty set.

Definition. The map f is *summable* over $N^{(p-1)}$ (or *the family* $\{f(a) : a \in N^{(p-1)}\}$ *is summable in* ℓ_r) if there exists a $g \in \ell_r$ with the following property: given $\epsilon > 0$ there is an $A \in F$ for which

$$\|I(B) - g\| < \epsilon$$

whenever $B \in F$ with $A \subset B$.

In connection with the proof of Theorem 6, we mention that the condition that f is summable on $N^{(p-1)}$ for $p = 2$ is equivalent to the proposition that the series

$$\sum_{a=0}^{\infty} f(a)$$

is *unconditionally convergent*, meaning that every rearrangement converges in ℓ_r.

The theorem given below is more general than needed in Section II, where only the $r = 1$ case is used. (The more general result may be of interest in other applications. Its proof is a direct extension of the proof for $r = 1$.)

Theorem 7. f is summable over $N^{(p-1)}$ if and only if

$$\sup_{B \in F} \|I(B)\| < \infty.$$

Proof

As mentioned earlier, proofs are omitted in this version of the paper.

REFERENCES

[1] I.W. Sandberg, "Criteria for the Stability of p-Linear Maps and p-Powers," to appear in *Nonlinear Analysis, Theory, Methods and Applications* (see also "Criteria for the Stability of Multilinear Systems," *Proceedings of the IEEE International Symposium on Circuits and Systems*, Portland Oregon, May 8–10, 1989).

[2] E.W. Kamen, "On the Relationship Between Bilinear Maps and Linear Two-dimensional Maps," Nonlinear Analysis, Theory, Methods and Applications, vol. 3, no. 4, pp. 467–481, 1979.

[3] I.W. Sandberg, "Multilinear Maps and Uniform Boundedness," IEEE Trans. Circuits and Systems, vol. CAS-32, no. 4, April 1985.

[4] I.W. Sandberg, "A Perspective on System Theory," IEEE Transactions on Circuits and Systems, vol. CAS–31, pp. 88–103, January 1984.

[5] S. Boyd, L.O. Chua, and C.A. Desoer, "Analytical Foundations of Volterra Series," IMA Journal of Mathematical Control and Information, vol. 1, pp. 243–282, 1984.

[6] M. Fliess, M. Lamnabhi, and F. Lamnabhi-Lagarrigue, "An Algebraic Approach to Nonlinear Functional Expansions," IEEE Trans. Circuits Syst, vol. CAS-30, pp. 554–570, Aug. 1983.

[7] W.J. Rugh, *Nonlinear System Theory*, Baltimore, MA, The John Hopkins University Press, 1981.

[8] R.J.P. de Figueiredo and T.A.W. Dwyer, III, "A Best Approximation Framework and Implementation for Simulation of Large-scale Nonlinear Systems," IEEE Trans. Circuits Syst., vol. CAS-27, pp. 1005–1014, Nov. 1980.

[9] D.D. Weiner and J.F. Spina, *Sinusoidal Analysis and Modeling of Weakly Nonlinear Circuits*, New York, Van Nostrand, 1980.

[10] I. J. Maddox, *Infinite Matrices of Operators*, Springer-Verlag, Berlin, 1980.

[11] M.S. Macphail, "Absolute and Unconditional Convergence," Bull. Am. Math. Soc., vol. 53, pp. 121–123, 1947.

[12] A.J.E.M. Janssen and P. van der Steen, *Integration Theory*, Springer-Verlag, Berlin, 1984.

ROBUST PERFORMANCE IN THE PRESENCE OF ADDITIVE AND MULTIPLICATIVE UNCERTAINTIES[1]

JIE CHEN AND JAMES S. FREUDENBERG
The Department of Electrical Engineering
and Computer Science
The University of Michigan
Ann Arbor, MI 48109

ABSTRACT

The robust performance problem of maintaining the output sensitivity function small despite the presence of multiplicative uncertainty at the plant input as well as additive uncertainty is treated as an equivalent robust stability problem by adding an additional divisive uncertainty at the plant output, and analyzed using bounds upon the structured singular value. The additive uncertainty can interact with the multiplicative uncertainty at the plant input, just as the multiplicative uncertainty itself does with the divisive uncertainty at the plant output, to produce robustness problems and thus impose stringent constraints upon the size and shape of the closed loop transfer functions. It is shown in this paper that the design guidelines developed in [9], [10], aiming at preventing undesirable interactions between the multiplicative and divisive uncertainty, are also useful in coping with the design difficulties due to interactions between the additive and multiplicative uncertainty.

1. INTRODUCTION

An important objective of feedback in control system design is to achieve satisfactory stability and performance properties despite the presence of errors in the plant model. Performance robustness problems have recently received considerable interest (c.f. [1], [3], [8], [9], [11], [14], [15], [16], to name just a few.). One approach to such a problem is to reformulate it as an equivalent robust stability problem, which may then be analyzed using the structured singular value [5], [8].

In studying robust performance, it has been instructive to consider certain *canonical* problems which expose inherent limitations posed by *ill-conditioned* plants [9], [13], [14], [15]. By ill-conditioned, we mean that the gain of the plant, at a frequency of interest, exhibits strong directional properties: some input signals will be amplified much more than will others. In [9], [10], the problem of maintaining output sensitivity robust against unstructured multiplicative uncertainty at the plant input was considered. The analysis in these papers led to design guidelines useful for coping with an ill-conditioned plant. Briefly, these guidelines are that the loop transfer function should *also* be ill-conditioned, and should possess directionality properties compatible with those of the uncompensated plant. A natural question is whether these guidelines are themselves "robust" against additional sources of modelling error not explicitly considered in [9], [10]. Such a worry is not without motivation. For example, if we require performance at the plant output to be robust with respect to uncertainty at the plant output, the resulting design guidelines

---

[1] This research is supported in part by AFOSR under Contract No. F33615-88-C-3601 and in part by NSF under Grant ECS-8857510

will be drastically different from those needed to achieve output performance robust with respect to uncertainty at the plant input.

In the present paper we consider the robust performance problem studied in [9], [10] with an *additional* source of modelling error which we take to be an unstructured additive uncertainty. We shall show that the design guidelines developed in [9], [10] remain *qualitatively* unchanged when the additive uncertainty is also taken into account.

The rest of this paper is organized as follows. In Section 2, we introduce the feedback system with modelling errors considered throughout this paper, and the robust performance problem is posed as an equivalent robust stability problem. In Section 3, we present several useful bounds upon the general structured singular value with respect to two and three sources of uncertainty. As in [9], [10], [3], these bounds serve as the foundation for our analysis. In Section 4, the stability robustness problem is analyzed using the bounds upon the structured singular value and insights are obtained by investigating the interrelations among the parameters in terms of which the bounds are expressed. The paper is concluded in Section 5.

2. THE ROBUST PERFORMANCE PROBLEM

Consider the linear time-invariant feedback system depicted in Figure 1, where $P(s)$ denotes the transfer function matrix of the plant model, and $F(s)$ denotes the transfer function matrix of the compensator. We shall assume[2] that the plant has n inputs and outputs, and that $det P(s) \not\equiv 0$. Define the *input open loop transfer function, sensitivity function, and complementary sensitivity function*:

$$L_I(s) = F(s)P(s),\ S_I(s) = [I + L_I(s)]^{-1},\ T_I(s) = L_I(s)[I + L_I(s)]^{-1}, \tag{2.1}$$

and the *output open loop transfer function, sensitivity function, and complementary sensitivity function*:

$$L_O(s) = P(s)F(s),\ S_O(s) = [I + L_O(s)]^{-1},\ T_O(s) = L_O(s)[I + L_O(s)]^{-1}. \tag{2.2}$$

Our nominal performance goal will be achieved if the output sensitivity function satisfies a frequency dependent bound upon its magnitude, and the *robust performance* goal will be satisfied if this bound holds despite the presence of plant modelling uncertainty. In [9], [10], useful design guidelines were developed for a canonical problem which concerns achieving robust performance against multiplicative uncertainty at the plant input. The issue of concern in the present paper is to determine whether these guidelines must be altered qualititatively when additive uncertainty is also present. As discussed in [8], such a performance robustness problem can be treated as an equivalent robust stability problem against the actual sources of plant uncertainty plus an additional source of *fictitious* uncertainty used to represent the performance specification. In this regard, the present problem is equivalent to the problem of maintaining stability robust against plant uncertainty in the form of

$$P_\delta(s) = [I + r_2(s)\Delta_2(s)]^{-1}[P(s) + r_3(s)\Delta_3(s)][I + r_1(s)\Delta_1(s)] \tag{2.3}$$

---

[2] Our results extend to nonsquare plants that satisfy the relevant assumption of left or right invertibility.

which corresponds to the block diagram depicted in Figure 2. Here $r_i(s)$, $i = 1, 2, 3$, are stable, minimum phase scalar weighting functions, and $\Delta_i(s)$, $i = 1, 2, 3$, lie in the set

$$\mathbf{D}_\gamma = \{\Delta_i(s) : \Delta_i(s)\, \mathit{is\ stable\ and}\, \bar{\sigma}[\Delta_i(j\omega)] < \gamma,\ \forall\omega\,\} \tag{2.4}$$

Note that $\Delta_1(s)$ and $\Delta_3(s)$ are actual sources of plant uncertainty, while $\Delta_2(s)$ is the fictitious uncertainty used to represent the performance specification. The weighting functions are used to describe how the uncertainties and performance requirement vary with frequency.

We shall, for the purpose of brevity, discuss only this stability robustness problem in detail, and leave interpretations of the corresponding performance robustness problem to be developed by the reader. Notice that setting $\Delta_3(s) = 0$ leads to the canonical performance robustness problem studied in [9], [10], and the latter problem was interpreted as a stability robustness problem against uncertainties occuring simultaneously at both the plant input and output.

3. BOUNDS ON THE STRUCTURED SINGULAR VALUE

The first step in the structured singular value (SSV) analysis is to rearrange the block diagram of the feedback system into the form shown in Figure 3. The uncertainty matrix $\Delta(s)$ is assumed to lie in the set

$$\mathbf{\Delta} = \left\{\, \Delta(s) :\ \Delta(s) = diag[\Delta_1(s), \Delta_2(s), \cdots, \Delta_k(s)],\ \Delta_i \in \mathbf{C}^{\mathbf{k_i} \times \mathbf{k_i}}\ \mathit{and\ stable} \right\} \tag{3.1}$$

It is often convenient to assume that system uncertainties have been scaled to satisfy the upper bound $\bar{\sigma}[\Delta_i(j\omega)] \leq \gamma$, $\forall\omega$, $\forall i$, and to introduce the set $\mathbf{\Delta}_\gamma = \{\ \Delta(s) :\ \Delta(s) \in \mathbf{\Delta}\ \mathit{and}\ \bar{\sigma}[\Delta(j\omega)] \leq \gamma,\ \forall\omega\ \}$. The interconnection matrix

$$M(s) = [M_{ij}(s)]\ \ i, j = 1, 2, \cdots, k, \quad M_{ij}(s) \in \mathbf{C}^{\mathbf{k_i} \times \mathbf{k_j}} \tag{3.2}$$

is stable if the feedback system is nominally stable. At each frequency, the structured singular value, denoted $\mu[M]$, is defined[3] by [5], [8][4]

$$\mu[M] = \begin{cases} 0 & det[I + M\Delta] \neq 0\ \forall\Delta \in \mathbf{\Delta} \\ 1/(\min_{\Delta\in\mathbf{\Delta}}\{\bar{\sigma}[\Delta] : det[I + M\Delta] = 0\}) & otherwise \end{cases} \tag{3.3}$$

Hence, by definition, the reciprocal of the structured singular value is a frequency dependent stability margin with respect to uncertainty of the form (3.1), and a nominally stable feedback system will be robustly stable against all uncertainty in the set $\mathbf{\Delta}_\gamma$ if and only if $\mu[M(j\omega)] < 1/\gamma$, $\forall\omega$.

We now summarize the bounds upon the structured singular value with respect to two and three sources of uncertainty; readers are referred to [2] for further details. Let $\sigma_{ij} = \bar{\sigma}[M_{ij}]$, and define the *interaction parameters* [2], [3]

$$\begin{aligned} &\mu(i, j) = \sqrt{\sigma_{ij}\sigma_{ji}}\,, \quad \forall i \neq j, \\ &\mu(i, j, k) = (\sigma_{ij}\sigma_{jk}\sigma_{ki})^{1/3}\,, \quad \forall i \neq j \neq k, \\ &\mu_1 = \max_i \sigma_{ii}, \\ &\mu_2 = \max_{i,j} \mu(i, j), \\ &\mu_3 = \max_{i,j,k} \mu(i, j, k). \end{aligned} \tag{3.4}$$

---

[3] Throughout this paper, we suppress dependence upon frequency whenever appropriate.

[4] By a mild abuse of notation, the symbol $\mathbf{\Delta}$ will occasionally be used to denote the set of constant complex matrices with block diagonal structure (3.1).

The interaction parameter $\mu(i,j)$ determines to what extent the effects of the two uncertainties Δ_i and Δ_j can interact to produce robustness difficulties. The robust stability problem obtained by setting $\Delta_k = 0$ can be studied using structured singular value techniques applied to the matrix obtained by deleting the kth block row and column of M. Denote this matrix by $\mu[M|_{i,j}]$.

Proposition 3.1 ([2]) *Choose $i \neq j \neq k$. Then*

$$max\ \{\sigma_{ii},\ \sigma_{jj},\ \mu(i,j)\} \leq \mu[M|_{i,j}] \leq max\ \{\sigma_{ii},\ \sigma_{jj}\} + \mu(i,j) \tag{3.5}$$

$\nabla\nabla\nabla$

We next consider the combined effects of all three uncertainties.

Proposition 3.2 ([2]) *Consider μ_1, μ_2, and μ_3 defined in (3.4). Then*
(i)

$$\mu[M] \geq max\ \{\mu_1, \mu_2, \mu_3\} \tag{3.6}$$

(ii) Suppose that $\mu_3 \neq 0$. If $\mu_2/\mu_3 \leq 1$, then

$$\mu[M] \leq \mu_1 + \mu_2 + \mu_3 \tag{3.7}$$

(iii) Suppose $\mu_2 \neq 0$. If $\mu_3/\mu_2 \leq 1$, then

$$\mu[M] \leq \mu_1 + 2.74\mu_2 \tag{3.8}$$

$\nabla\nabla\nabla$

It follows that forcing the interaction parameters to be small is *necessary* to achieve stability robustness. The upper bounds above show that, despite the gap between the upper and lower bounds, this strategy also tends to be *sufficient*. Further interpretations of these parameters are found in [4]. For our purpose, we next turn to an analysis of the interrelations among the blocks of M (and thus the interaction parameters) for our specific robustness problem.

4. SSV ANALYSIS OF THE STABILITY ROBUSTNESS PROBLEM

We now formulate the stability robustness problem posed in Section 2 as a three-block structured singular value problem. Our first task is to find the corresponding interconnection matrix. Manipulating the block diagram in Figure 2 yields:

$$M = \begin{bmatrix} r_1T_I & -r_1T_IP^{-1} & r_1T_IP^{-1} \\ -r_2S_OP & r_2S_O & -r_2S_O \\ -r_3S_I & -r_3T_IP^{-1} & r_3T_IP^{-1} \end{bmatrix} \tag{4.1}$$

Next, we need to identify the parameters defined in (3.4), which are given as follows.

$$\begin{aligned} \mu_1 &= max\ \{\bar{\sigma}[r_1T_I],\ \bar{\sigma}[r_2S_O],\ \bar{\sigma}[r_3T_IP^{-1}]\} \\ \mu(1,2) &= \sqrt{\bar{\sigma}[r_2S_OP]\bar{\sigma}[r_1T_IP^{-1}]} \\ \mu(1,3) &= \sqrt{\bar{\sigma}[r_3S_I]\bar{\sigma}[r_1T_IP^{-1}]} \\ \mu(2,3) &= \sqrt{\bar{\sigma}[r_2S_O]\bar{\sigma}[r_3T_IP^{-1}]} \\ \mu(1,2,3) &= (\bar{\sigma}[r_1T_IP^{-1}]\bar{\sigma}[r_2S_O]\bar{\sigma}[r_3S_I])^{1/3} \\ \mu(1,3,2) &= (\bar{\sigma}[r_1T_IP^{-1}]\bar{\sigma}[r_2S_OP]\bar{\sigma}[r_3T_IP^{-1}])^{1/3} \end{aligned} \tag{4.2}$$

Note that if the feedback system is nominally stable, then it will be robustly stable against $\Delta_1 \in \mathbf{D}_\gamma$ alone if and only if $\bar{\sigma}[r_1(j\omega)T_I(j\omega)] < 1/\gamma$, $\forall\omega$ [7]. Similar statements apply to Δ_2 and $\bar{\sigma}[r_2(j\omega)S_O(j\omega)]$, and to Δ_3 and $\bar{\sigma}[r_3(j\omega)T_I(j\omega)P^{-1}(j\omega)]$. Hence the parameter μ_1 summarizes stability robustness with respect to each of the three uncertainty sources *individually.* Assuming that $\bar{\sigma}[r_1(j\omega)T_I(j\omega)] < 1/\gamma$, then it is a simple exercise to show that

$$\bar{\sigma}[r_3(j\omega)T_I(j\omega)P^{-1}(j\omega)] < \frac{\mid r_3(j\omega) \mid}{\gamma \mid r_1(j\omega) \mid \underline{\sigma}[P(j\omega)]} \tag{4.3}$$

It follows that achieving stability robustness against Δ_1 alone *automatically* guarantees a certain, albeit possibly smaller, level of stability robustness against Δ_3 alone.

Next, we analyze the effects of uncertainties taken two at a time. The effect of interactions between Δ_1 and Δ_2 is quantified by $\mu[M|_{1,2}]$ and has been studied in [9], [10]. Applying the bound (3.5) to this problem, it follows that the parameter $\mu(1,2)$ determines the extent to which these two uncertainties can interact to cause stability robustness difficulties even if the system is robustly stable against Δ_1 and Δ_2 individually. To prevent this undesirable effect imposes a *nontrivial* problem, for merely requiring $\bar{\sigma}[r_1T_I]$ and $\bar{\sigma}[r_2S_O]$ to be small does not imply that $\mu(1,2)$ is also small. To this point, let the plant *condition number* [12] be defined by $\kappa[P(j\omega)] = \bar{\sigma}[P(j\omega)]/\underline{\sigma}[P(j\omega)]$.

Proposition 4.1 ([9]) *Let $\mu(1,2)$ be defined in (4.2). Then, at each frequency,*

$$\mu(1,2) \le \sqrt{\kappa[P]} \cdot \sqrt{\bar{\sigma}[r_1T_I]\bar{\sigma}[r_2S_O]} \tag{4.4}$$

▽▽▽

Since the bound (4.4) may be tight [9], it follows that stability robustness against Δ_1 and Δ_2 together is potentially much worse than that against each of these two *individually* at frequencies for which the plant is ill-conditioned ($\kappa[P] \gg 1$). It was shown in [9], [10] that requiring $\mu(1,2)$ to be small imposes more stringent constraints upon the closed loop transfer functions than those due to the stability robustness requirements with respect to Δ_1 or Δ_2 *alone.* Consider the *singular value decomposition* [12] of the plant transfer function, and let the right singular vectors corresponding to the largest and smallest singular values be denoted z_1 and z_n. Then, it was shown that $\mu(1,2) < 1/\gamma$ *only if* the closed loop transfer functions satisfy the bound

$$\|z_1^H S_I\| \cdot \|T_I z_n\| < \frac{1}{\gamma \mid r_1 r_2 \mid \kappa[P]} \tag{4.5}$$

As discussed in [7], [8], it is typical to select the weighting functions r_1 and r_2 so that $\mid r_1(j\omega) \mid$ is large only at high frequencies and $\mid r_2(j\omega) \mid$ is large only at low frequencies. For later reference, note that requiring $\bar{\sigma}[r_1(j\omega)T_I(j\omega)]$ to be small at a frequency for which $\mid r_1(j\omega) \mid \gg 1$ forces $S_I(j\omega) \approx I$ at that frequency. Thus at frequencies where these conditions hold, it follows that the bound (4.5) reduces to requiring

$$\|T_I z_n\| < \frac{1}{\gamma \mid r_1 r_2 \mid \kappa[P]} \tag{4.6}$$

Similarly, we analyze the combined effects of uncertainties Δ_2 and Δ_3.

Proposition 4.2 *Let $\mu(2,3)$, μ_1 be defined in (4.2). Then, at each frequency,*

$$\mu[M|_{2,3}] \le 2\mu_1 \tag{4.7}$$

▽▽▽

Proof: By definition,

$$\mu(2,3) = \sqrt{\bar{\sigma}[r_2 S_O]\bar{\sigma}[r_3 T_I P^{-1}]} \leq \sqrt{\mu_1^2}$$

The proof now follows from Proposition 3.1.

▽▽▽

Proposition 4.2 reveals that, unlike the problem of maintaining stability robustness with respect to Δ_1 and Δ_2, a feedback design which is robust against Δ_2 and Δ_3 *alone* will tend to be robust with respect to these two uncertainties taken together, independent of the size of the plant condition number.

Lastly, we consider the combined effects of uncertainties Δ_1 and Δ_3.

Proposition 4.3 *Let $\mu(1,3)$, μ_1 be defined in (4.2). Then, at each frequency,*

$$\mu(1,3) \leq \sqrt{\mu_1(\mid r_1 \mid + \mu_1)} \tag{4.8}$$

$$\mu(1,3) \geq \sqrt{\bar{\sigma}[r_3 T_I P^{-1}]\,(\mid r_1 \mid - \bar{\sigma}[r_1 T_I])} \tag{4.9}$$

▽▽▽

Proof: It follows from definition that

$$\begin{aligned} \mu(1,3) &= \sqrt{\bar{\sigma}[r_1 T_I P^{-1}]\bar{\sigma}[r_3 S_I]} \\ &= \sqrt{\bar{\sigma}[r_3 T_I P^{-1}]\bar{\sigma}[r_1 (I + T_I)]} \end{aligned}$$

The proof is now completed by applying the triangle inequality.

▽▽▽

The bounds in (4.8-9) exhibit that stability robustness with respect to Δ_1 and Δ_3 *together* is potentially much worse than that with respect to either of these uncertainties alone at any frequency for which $\mid r_1(j\omega) \mid \gg 1$. Analyses similar to those performed in [9], [10] give rise to

$$\mu(1,3) \geq \sqrt{\mid r_3 \mid \cdot \|T_I z_n\| \cdot (1/\underline{\sigma}[P])(\mid r_1 \mid - \bar{\sigma}[r_1 T_I])} \tag{4.10}$$

Hence, it follows that if $\bar{\sigma}[r_1 T_I]$ is small, and $\mid r_1 \mid \gg 1$, then maintaining $\mu(1,3) < 1/\gamma$ requires that the input complementary sensitivity function satisfy the requirement

$$\|T_I z_n\| < \frac{\underline{\sigma}[P]}{\gamma \mid r_1 r_3 \mid} \tag{4.11}$$

Note that this requirement imposes a more stringent constraint than satisfying $\|T_I z_n\| < \underline{\sigma}[P]/\gamma \mid r_3 \mid$, which is necessary in order to satisfy $\bar{\sigma}[r_3 T_I P^{-1}] < 1/\gamma$. For our purpose, it is important to notice the similarity between (4.6) and (4.11); each displays the need of an additional requirement upon the shape and size of T_I at frequencies for which $\mid r_1 \mid \gg 1$. The relevant observation is that the qualitative design guidelines developed in [9], [10] for maintaining $\mu(1,2)$ small also hold for the problem of maintaining $\mu(1,3)$ small. In particular, if $\mu(1,2) < 1/\gamma$, then

$$\mu(1,3) < \sqrt{\frac{\mid r_3 \mid}{\gamma \mid r_2 \mid \underline{\sigma}[P]}} \tag{4.12}$$

and it follows from Proposition 4.1 that achieving stability robustness with respect to Δ_1 and Δ_2 together also tends to guarantee a certain smaller level of stability robustness with respect to the combined effects of Δ_1 and Δ_3.

Finally, we need to consider the simultaneous effects of *all* three uncertainties. Recall that these effects are determined by the parameters $\mu(1,2,3)$ and $\mu(1,3,2)$. Fortunately, for this particular problem, requiring μ_1 and μ_2 to be small forces μ_3 small as well.

Proposition 4.4 *Let μ_1, μ_2, and μ_3 be defined in (4.2). Then, at each frequency,*

$$\mu_3 \leq max \; \{ \mu_1, \, \mu_2 \} \tag{4.13}$$

▽▽▽

Proof: First, it is straightforward to show

$$\mu(1,3,2) \leq max \; \{ \mu_1, \, \mu_2 \}$$

since

$$\begin{aligned} \mu(1,3,2) &= \left(\bar{\sigma}[r_1 T_I P^{-1}]\bar{\sigma}[r_3 T_I P^{-1}]\bar{\sigma}[r_2 S_O P]\right)^{1/3} \\ &= \left(\mu^2(1,2)\bar{\sigma}[r_3 T_I P^{-1}]\right)^{1/3} \\ &\leq \left(\mu_2^2 \cdot \mu_1\right)^{1/3} \end{aligned}$$

Next, following the same line establishes

$$\mu(1,2,3) \leq max \; \{ \mu_1, \, \mu_2 \}$$

since

$$\begin{aligned} \mu(1,2,3) &= \left(\bar{\sigma}[r_1 T_I P^{-1}]\bar{\sigma}[r_3 S_I]\bar{\sigma}[r_2 S_O]\right)^{1/3} \\ &= \left(\mu^2(1,3)\bar{\sigma}[r_2 S_O]\right)^{1/3} \\ &\leq \left(\mu_2^2 \cdot \mu_1\right)^{1/3} \end{aligned}$$

This completes the proof.

▽▽▽

To summarize, we observe that for the specific problem we study in this paper, the uncertainty Δ_1 can interact with either Δ_2 or Δ_3 to produce stability robustness problems, and to prevent these problems imposes stringent design constraints upon the closed loop transfer functions. Fortunately, the design guidelines developed in [9], [10], aiming at preventing the undesirable interactions between Δ_1 and Δ_2, are also useful in preventing the undesirable interactions between Δ_1 and Δ_3. Notice that achieving stability robustness against the effects of these two pairs of uncertainties tends to prevent the overall robustness from being poor. Hence, the guidelines in [9], [10] remain qualitatively valid to the present problem.

5. CONCLUDING REMARKS

In this paper we have studied robust performance of a feedback system in the presence of both multiplicative uncertainty at the plant input as well as additive uncertainty. This problem was formulated as an equivalent robust stability problem by inserting an additional divisive uncertainty at the plant output, and analyzed using the structured singular value methodology with respect to three sources of modelling error. By utilizing bounds upon the structured singular value, we showed that the multiplicative uncertainty can interact with either the additive uncertainty or the divisive uncertainty to produce robustness problems, and to prevent these problems imposes stringent design constraints upon the closed loop transfer functions. Fortunately, these uncertainties interact in such a way that achieving stability robustness with respect to the combined effects of multiplicative and divisive uncertainty also tends to guarantee certain level of stability robustness with respect to the combined effects of additive and multiplicative uncertainty, and hence further insure certain level of stability robustness with respect to the combined effects of all three uncertainties. Consequently, the design guidelines developed in [9], [10], without taking the additive uncertainty into consideration, remain qualitatively valid even if the additive uncertainty is indeed present.

REFERENCES

[1] Bird, J. S., and Francis, B. A., "On the Robust Disturbance Attenuation Problem," *Proc. 25th Conf. on Decision and Contr.*, Dec. 1986.

[2] Chen, J., and Freudenberg, J. S., "Bounds on the Three-Block Structured Singular Value," *7th Annual OSU Control Workshop*, Columbus, OH, April 1989, also submitted for publication.

[3] Chen, J., and Freudenberg, J. S., "Robust Performance with Respect to Diagonal Input Uncertainty," To appear, *Proc. 28th Conf. on Decision and Contr.*, Dec. 1989, also to be submitted to *IEEE Trans. Aut. Contr.*.

[4] Chen, J., and Freudenberg, J. S., "Robustness Properties of Cyclic Iinterconnection Matrices," Tech. Rept., The University of Michigan, Ann Arbor, MI, 1989.

[5] Doyle, J. C., "Analysis of Feedback Systems with Structured Uncertainties," *IEE Proc.*, vol. 129, Pt.D, no. 6, Nov. 1982.

[6] Doyle, J. C., "Structured Uncertainty in Control System Design," *Proc. 24th Conf. on Decision and Contr.*, Dec. 1985.

[7] Doyle, J. C., and Stein, G., "Multivariable Feedback Design: Concept for a Classical/Modern Synthesis," *IEEE Trans. Aut. Contr.*, vol. AC-26, Feb. 1981.

[8] Doyle, J. C., Wall, J. E., and Stein, G., "Performance and Robustness Analysis for Structured Uncertainty," *Proc. 21st IEEE Conf. on Decision and Contr.*, Dec. 1982.

[9] Freudenberg, J. S., "Analysis and Design for Ill-Conditioned Plants, Part 1: Lower Bounds on the Structured Singular Value," *Int. J. Contr.*, Vol. 49, No. 3, March 1989.

[10] Freudenberg, J. S., "Analysis and Design for Ill-Conditioned Plants, Part 2: Directionally Uniform Weightings and an Example," *Int. J. Contr.*, Vol. 49, No. 3, March 1989.

[11] Freudenberg, J. S., and Looze, D. P., *Frequency Domain Properties of Scalar and Multivariable Feedback Systems*, Springer-Verlag Lecture Notes in Control and Information Sciences, 1988.

[12] Golub, G. H., and Van Loan, C. F., *Matrix Computations*, Baltimore, MD, Johns Hopkins Univ. Press, 1983.

[13] Nett, C. N., and Manousiouthakis, V., "Euclidean Condition and Block Relative Gain: Connections, Conjectures, and Clarifications," *IEEE Trans. Aut. Contr.*, Vol. AC-32, No. 5, May 1987.

[14] Skogestad, S., Morari, M., and Doyle, J. C., "Robust Control of Ill-Conditioned Plants: High-Purity Distillation," *IEEE Trans. Aut. Contr.*, Vol. AC-33, No. 12, Dec. 1988.

[15] Stein, G., "Beyond Singular Values and Loop Shapes," Report No. LIDS-P-1504, MIT, Cambridge, MA, August 1985.

[16] Stein, G., *Lecture Notes of Workshop on Robust Multivariable Control*, Minneapolis, MN, June 1987.

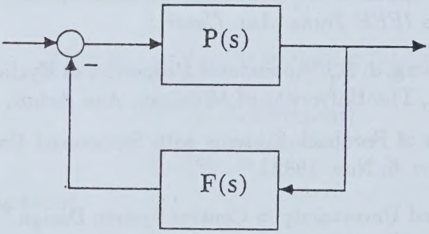


Figure 1. Block Diagram of Feedback Systems

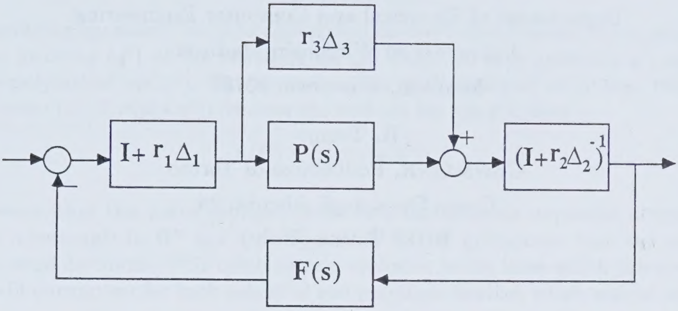

r3Δ3
I+ r1Δ1
P(s)
+
(I+r2Δ2)-1
F(s)

Figure 2. Block Diagram of Robust Performance Problem

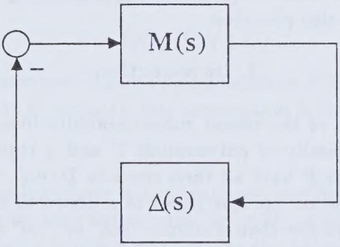


Figure 3. Block diagram for structured singular value analysis

On the Spectral Set for a Family of Polynomials

B. R. Barmish
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, Wisconsin 53706

R. Tempo
CENS-CNR, Politecnico di Torino
Corso Duca degli Abruzzi 24
10129 Torino (Italy)

Abstract

Given a family of real n-th order polynomials $\mathcal{P}$, the *spectral set* of $\mathcal{P}$ is given by

$$\sigma(\mathcal{P}) \doteq \{z \in \mathbf{C} : p(z) = 0 \ \text{ for some } \ p(\cdot) \in \mathcal{P}\}.$$

The focal point of this paper is the problem of generating the spectral set for two different families of polynomials—spherical families and polytopic families. For the spherical case, a *closed form equation* describing the boundary of the spectral set is obtained. In the polytopic case, a a computationally feasible technique is given which requires only a two dimensional gridding of a bounded subset of the complex plane rather than a high dimensional gridding of the parameter set. Numerical examples are also provided.

I. Introduction

Generally speaking, much of the recent robust stability literature is concerned with the following problem: Given a family of polynomials $\mathcal{P}$ and a region $\mathcal{D}$ in the complex plane, determine if all polynomials in $\mathcal{P}$ have all their roots in $\mathcal{D}$; e.g., see Barmish [1] for a review of this literature. The motivation for this paper is a desire to know something more about the "root distribution" of $\mathcal{P}$ rather then a simple "yes" or "no" answer whether all roots are in $\mathcal{D}$. From a control systems point of view, knowing the root distribution makes it possible to analyze robust performance—not only robust stability. This leads to the problem of the spectral set generation.

More formally, when all polynomials in $\mathcal{P}$ have the same degree, the *spectral set* of $\mathcal{P}$ is defined as

$$\sigma(\mathcal{P}) \doteq \{z \in \mathbf{C} : p(z) = 0 \ \text{ for some } \ p(\cdot) \in \mathcal{P}\}. \tag{1}$$

Note that once $\sigma(\mathcal{P})$ has been found, the robust stability problem is solved for all possible $\mathcal{D}$ regions. That is, given an arbitrary $\mathcal{D}$ region, robust stability is guaranteed if and only if

$$\sigma(\mathcal{P}) \subseteq \mathcal{D}. \tag{2}$$

Hence, the problem of spectral set generation can be viewed as a generalization of the $\mathcal{D}$-stability problem.

The main objective of the paper is to describe viable computational techniques to generate $\sigma(\mathcal{P})$ for two types of polynomial families $\mathcal{P}$— spherical families and polytopic families. By "viable," we mean that gridding the so-called *perturbation set* is avoided. To elaborate on this point, suppose that the family $\mathcal{P}$ is obtained as some *perturbation vector* q ranges over a given *bounding set* $Q \subseteq \mathbf{R}^l$. That is, associated with each $q \in Q$ is a polynomial $p(s, q)$. Therefore, we write

$$\mathcal{P} = \{p(s,q) : q \in Q\}. \tag{3}$$

Now, the "gridding approach" leads to approximation of $\sigma(\mathcal{P})$ as follows: First, generate some suitably fine gridding $\{q^i\}$ of the bounding set Q. Then, for each gridpoint q^i, compute the roots of the polynomial $p(s, q^i)$. The approximation of the spectral set is then the collection of all such roots; i.e., if $\sigma(p(s, q^i))$ denotes the root set for $p(s, q^i)$, then

$$\sigma(\mathcal{P}) \approx \bigcup_i \sigma(p(s,q^i)). \tag{4}$$

Notice, however, that this naive approach leads to a combinatoric explosion in computation; e.g., if Q is a rectangle in $\mathbf{R}^n$ and each edge of Q has N gridpoints, then the roots of N^n polynomials must be found. This combinatoric explosion is the issue which the techniques in this paper will circumvent for both spherical and polytopic families which will be now defined.

II. Spectral Set Problem for Spherical and Polytopic Families

The Spherical Problem: Given a radius $r > 0$, a *spherical family* of polynomials $\mathcal{P}$ is described by

$$p(s,q) \doteq p_0(s) + \sum_{i=0}^{n} q_i s^i \tag{5}$$

where $p_0(s)$ is a fixed n-th order polynomial and q ranges in the sphere

$$q \in Q_r \doteq \left\{q : q^T q \le r\right\}. \tag{6}$$

Hence, we obtain the family of polynomials

$$\mathcal{P} = \{p(s,q) : q \in Q\}. \tag{7}$$

Throughout this paper it is assumed that all polynomials in $\mathcal{P}$ have degree equal to n. Hence, if a_n denotes the coefficient of s^n in $p_0(s)$, this assumption is equivalent to

$$|a_n| > r. \tag{8}$$

Note that the degree requirement above is automatically satisfied whenever $p(s, q)$ is monic.

Remarks: It will be shown that the spectral set $\sigma(\mathcal{P})$ admits a closed form characterization. That is, we will construct a function $\Phi(z)$ having the following property: $z \in \sigma(\mathcal{P})$ if and only if

$$\Phi(z) \le 0. \tag{9}$$

This motivates a simple computational method for generating the boundary of $\sigma(\mathcal{P})$. At the end of the paper, the radius r is treated as a variable and we show that the robust stability results in [2]-[4] on spherical families of polynomials "fall out" quite naturally from the $\Phi(z)$ characterization of $\sigma(\mathcal{P})$.

The Polytopic Problem: In this case, $\mathcal{P}$ is a so-called *polytopic family* described by

$$\mathcal{P} = \operatorname{conv}\{p_1(s), p_2(s), \ldots, p_m(s)\} \tag{10}$$

where the $p_i(s)$ are fixed n-th order polynomials. This means that every polynomial $p(s)$ in $\mathcal{P}$ is a convex combination of the form

$$p(s) = \sum_{i=1}^{m} \lambda_i p_i(s) \tag{11}$$

where $\lambda_i \geq 0$ for $i = 1, 2, \ldots, m$ and

$$\sum_{i=1}^{m} \lambda_i = 1. \tag{12}$$

Again, all polynomials in $\mathcal{P}$ are assumed to have the degree equal to n. Letting a_{ni} denote the coefficient of s^n in $p_i(s)$, note that this degree requirement is satisfied if and only if all the a_{ni} have the same sign.

Remarks: For this polytopic case, we will present a computationally tractable method to compute $\sigma(\mathcal{P})$ which avoids the combinatoric explosion associated with an m dimensional gridding of the λ set. Instead, we only requires a two dimensional gridding of a bounded subset of the complex plane.

III. The Spherical Problem

Given a *candidate* point z in the complex plane, it follows that $z \in \sigma(\mathcal{P})$ if and only if

$$A(z)q = -\begin{bmatrix} Re\ p_0(z) \\ Im\ p_0(z) \end{bmatrix} \tag{13}$$

for some $q \in Q$ where

$$A(z) \doteq \begin{bmatrix} 1 & Re\ z & \cdots & Re\ z^n \\ 0 & Im\ z & \cdots & Im\ z^n \end{bmatrix}. \tag{14}$$

Equivalently, letting

$$\vec{p}_0(z) \doteq \begin{bmatrix} Re\ p_0(z) \\ Im\ p_0(z) \end{bmatrix}, \tag{15}$$

$z \in \sigma(\mathcal{P})$ exists if and only if the minimum norm solution $q(z)$ of (13) satisfies

$$q^T(z)q(z) \leq r^2. \tag{16}$$

We consider two cases.

Case 1: $Im\ z \neq 0$. Then,

$$\text{rank} \ \ A(z) = 2 \tag{17}$$

and we obtain a minimum norm solution

$$q(z) = -A^T(z)\left(A(z)A^T(z)\right)^{-1}\vec{p}_0(z) \leq r^2. \tag{18}$$

Hence, using (16), $z \in \sigma(\mathcal{P})$ if and only if

$$\vec{p}_0^{\,T}(z)\left(A(z)A^T(z)\right)^{-1}\vec{p}_0(z) \leq r^2. \tag{19}$$

Note that the *weighting matrix*

$$W(z) \doteq \left(A(z)A^T(z)\right)^{-1} \tag{20}$$

above is *universal* in the sense that is independent of the nominal polynom ial $p_0(s)$—it depends only on the degree n. In fact, since $A(z)A^T(z)$ is a 2×2 matrix, $W(z)$ is easily computed in closed form. Indeed, a lengthy but straightforward computation leads to the formula

$$W(z) = \frac{1}{\Delta(z)}\begin{bmatrix} \sum_{i=1}^{n}(Im\ z^i)^2 & -\sum_{i=1}^{n}(Re\ z^i)(Im\ z^i) \\ -\sum_{i=1}^{n}(Re\ z^i)(Im\ z^i) & \sum_{i=0}^{n}(Re\ z^i)^2 \end{bmatrix} \tag{21}$$

where

$$\Delta(z) \doteq \sum_{i=0}^{n}(Re\ z^i)^2 \cdot \sum_{i=1}^{n}(Im\ z^i)^2 - \left(\sum_{i=1}^{n}(Re\ z^i)(Im\ z^i)\right)^2 . \tag{22}$$

Case 2: $Im\ z = 0$. Now, noting that $\vec{p}_0(z)$ is real and $Re\ z^i = z^i$ for $i = 0, 1, 2, \ldots, n$ and $Im\ z^i = 0$ for $i = 1, 2, \ldots, n$, (14) becomes

$$\begin{bmatrix} 1 & z & z^2 & \cdots & z^n \end{bmatrix} q = p_0(z). \tag{23}$$

In this case, the minimum norm solution is

$$q(z) = -\frac{1}{\sum_{i=0}^{n} z^{2i}} \begin{bmatrix} 1 \\ z \\ z^2 \\ \vdots \\ z^n \end{bmatrix} p_0(z) \tag{24}$$

and by (16), it follows that $z \in \sigma(\mathcal{P})$ if and only if

$$p_0^2(z) - r^2 \sum_{i=0}^{n} z^{2i} \leq 0. \tag{25}$$

Therefore, the real roots of the polynomial on the left hand side of (25) describe the endpoints of intervals where $\sigma(\mathcal{P})$ intersects the real axis.

In conclusion, for both cases, $z \in \sigma(\mathcal{P})$ belongs to the spectral set if and only if an inequality of the form $\Phi(z) \leq 0$ is satisfied, where

$$\Phi(z) \doteq \begin{cases} \vec{p}_0^{\,T}(z) W(z) p_0(z) - r^2 & \text{if } Im\ z \neq 0; \\ p_0^2(z) - r^2 \sum_{i=0}^{n} z^{2i} & \text{if } Im\ z = 0. \end{cases} \tag{26}$$

Computational Generation of the Spectral Set: Since all points z in the boundary of the spectral set satisfy the simple equality $\Phi(z) = 0$, there are many computational methods available to generate $\sigma(\mathcal{P})$. For example, using 3-d graphics, one can simply plot $\Phi(z)$ and take a cross section at $\Phi(z) = 0$. Another possibility is to "track" the boundary $\sigma(\mathcal{P})$ in the complex plane; e.g., say z_t is the current iterate on the boundary of $\sigma(\mathcal{P})$ and $Im\ z_t \neq 0$. Let $\epsilon > 0$ denote a stepsize and take

$$d^t \doteq \begin{bmatrix} d_1^t \\ d_2^t \end{bmatrix} \tag{27}$$

to be the step direction which is a unit vector. Then enforcing the condition

$$\Phi(z_{t+1}) = 0 \tag{28}$$

and using a first order approximation to changes in $\Phi(z)$, we obtain

$$d^t = \left\{ \left(\frac{\partial \Phi(z)}{Re\ z}\Big|_{z_t}\right)^2 + \left(\frac{\partial \Phi(z)}{Im\ z}\Big|_{z_t}\right)^2 \right\}^{-1/2} \begin{bmatrix} -\frac{\partial \Phi(z)}{Im\ z}\Big|_{z_t} \\ \frac{\partial \Phi(z)}{Re\ z}\Big|_{z_t} \end{bmatrix}. \tag{29}$$

Note that the required partial derivatives above can either be supplied in closed form using the formula for $W(z)$ or approximated using some technique such as finite differences. To avoid buildup of numerical error, it is also worthwhile to project z_{t+1} onto the boundary of the set descibed by $\Phi(z) = 0$. This is accomplished via a simple line search; e.g., find $\lambda \in \mathbf{R}$ such that $\Phi(\lambda z_{t+1}) = 0$.

IV. Relation with Robust Stability Literature

Recalling Section 1, the robust stability problem can be viewed as a special case of the spectral set problem. That is, robust stability is guaranteed if and only if the spectral set does not cross the imaginary axis. Hence, to specialize the results above, we take $z = j\omega$. Then, for $\omega \neq 0$, the formulae describing $\sigma(\mathcal{P})$ greatly simplify. Namely, a straightforward calculation leads to the weighting matrix

$$W(j\omega) = \begin{bmatrix} (\sum_{i\ even} \omega^{2i})^{-1} & 0 \\ 0 & (\sum_{i\ odd} \omega^{2i})^{-1} \end{bmatrix}. \tag{30}$$

Now, using (25), it follows that if $\omega \neq 0$, then $j\omega \in \sigma(\mathcal{P})$ if and only if

$$\frac{(Re\ p(j\omega,0))^2}{\sum_{i\ even} \omega^{2i}} + \frac{(Im\ p(j\omega,0))^2}{\sum_{i\ odd} \omega^{2i}} \leq r^2. \tag{31}$$

For $\omega = 0$, the condition $0 \in \sigma(\mathcal{P})$ becomes

$$|p_0(0)| \leq r. \tag{32}$$

Equivalently, if a_0 denotes the zeroth order coefficient of $p_0(s)$, then (32) becomes

$$|a_0| \leq r. \tag{33}$$

In conclusion, robust stability is guaranteed if and if

$$|a_0| > r \tag{34}$$

and

$$\frac{(Re\ p(j\omega,0))^2}{\sum_{i\ even} \omega^{2i}} + \frac{(Im\ p(j\omega,0))^2}{\sum_{i\ odd} \omega^{2i}} > r^2 \tag{35}$$

for all $\omega \in \mathbf{R}$. These conditions amounts to a simplification of the results in [2] and are precisely the type of conditions obtained in [3] and [4].

V. The Polytopic Problem

In this case, we provide a simple technique for spectral set generation which avoids the combinatoric explosion discussed in Section I. Since this technique really amounts to a specialization of the so-called Robust Root Locus in [5], the exposition is greatly abbreviated. Indeed, given the polytopic family $\mathcal{P}$ and some fixed $z \in \mathbf{C}$, the so-called value set is given by

$$\Omega(z) \doteq \{p(z) : p(\cdot) \in \mathcal{P}\}. \tag{36}$$

In view of the polytopic description of $\mathcal{P}$, we have

$$\Omega(z) = \text{conv}\,\{p_1(z), p_2(z), \ldots, p_m(z)\} \tag{37}$$

and it follows that

$$\sigma(\mathcal{P}) = \{z \in \mathbf{C} : 0 \in \Omega(z)\}. \tag{38}$$

This suggests the algorithm below for generation of $\sigma(\mathcal{P})$.

Step 1: Compute some apriori bound $\sigma^+(\mathcal{P})$ (perhaps crude) for $\sigma(\mathcal{P})$. For example, writing

$$p_j(s) = \sum_{i=0}^{n} c_{ij} s^i \tag{39}$$

for $j = 1, 2, \ldots, n$, one such bound given in [6] is

$$\sigma^+(\mathcal{P}) = \{s : |s| \leq R\} \tag{40}$$

where

$$R = \frac{1 + \max\{c_{ij} : i = 0, 1, \ldots, n-1; j = 1, 2, \ldots, m\}}{\min\{c_{nj} : j = 1, 2, \ldots, m\}}. \tag{41}$$

Step 2: Generate a suitably fine gridding of $\sigma^+(\mathcal{P})$.

Step 3: For each grid point $z \in \sigma^+(\mathcal{P})$, check if $0 \in \Omega(z)$. If $0 \in \Omega(z)$, then we classify z as a point in $\sigma(\mathcal{P})$. If not, z is rejected. The net result of the procedure described above is a set of "clouds" representing $\sigma(\mathcal{P})$. Notice that checking the condition $0 \in \Omega(z)$ is quite simple because $\Omega(z)$ is a two dimensional convex polygon. This procedure is easily automated as illustrated in the example below. It is also worth noting that the problem of checking whether a point lies in a two-dimesional polygon is well known in computational geometry; e.g., see [7].

VI. Illustrative Examples

For the case of a spherical family of polynomials with nominal

$$p_0(s) = s^6 + 2s^5 + 3s^4 + 4s^3 + 5s^2 + 6s + 7 \tag{42}$$

and $r = 0.99$, the spectral set was computed using a boundary tracking algorithm as described in Section II. The results are shown in Figure 1.

For the polytopic case, we take an example considered previously in [8] in the context of robust stability of interval polynomials with respect to the unit circle. Here, however, we generate the entire spectral set. Indeed, we consider a so-called interval polynomial described by

$$p(s,q) = (5 + q_4)s^4 + (1 + q_3)s^3 + (1 + q_2)s^2 + (1 + q_1)s + (1 + q_0) \tag{43}$$

and $|q_i| \leq 0.85$ for $i = 0, 1, 2, 3, 4$. This leads to a polytope of polynomials $\mathcal{P}$ with 32 generators $p_i(s)$ obtained using the extreme values of the q_i. Then, using the two-dimensional gridding technique described in Section III, we generated the spectral set in Figure 2.

VII. Conclusion

In this paper, we provided methods for spectral set generation for both spherical and polytopic families of polynomials. In the spherical case, a closed form solution was exploited; in the polytopic case, the computational method involved a two dimensional gridding of a bounded subset of the complex plane.

Acknowledgements

This work was performed while B. R. Barmish was a Senior NATO Guest Fellow at CENS-CNR, Torino (Italy). The support of Professor A. R. Meo is gratefully ackowledged. Partial support of this project was also provided by the U.S. National Science Foundation under Grant No. ECS-8612948. The authors also express gratitude to Messrs. S. Dilip, S. Prichard and A. Takach for work on the numerical computation.

References

[1] B. R. Barmish, "New Tools for Robustness Analysis," *Proceedings of the IEEE Conference on Decision and Control*, Austin, 1988.

[2] C. B. Soh, C. S. Berger and K. P. Dabke, "On the Stability Properties of Polynomials with Perturbed Coefficients," *IEEE Transactions on Automatic Control*, AC-30, pp. 1033-1036, 1985.

[3] S. P. Bhattacharyya and H. Chapellat, "Stability Margin for Hur witz Polynomial," *Proceedings of the IEEE Conference on Decision and Control*, Austin, 1988.

[4] D. Hinrichsen and A. S. Pritchard, "New Robustness Results for Linear Systems under Real Perturbations," *Proceedings of the IEEE Conference on Decision and Control*, Austin, 1988.

[5] B. R. Barmish and R. Tempo, "The Robust Root Locus," *Proceedings of the IEEE Conference of Decision and Control*, Austin, 1988.

[6] M. Marden, "Geometry of Polynomials, *Mathematical Surveys*, no. 3, American Mathematical Society, Providence, 1966.

[7] F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.

[8] N. K. Bose, E. Jury and E. Zeheb, "On Robust Hurwitz and Schur Polynomials," *Proceedings of the IEEE Conference on Decision and Control*, Athens, 1987.

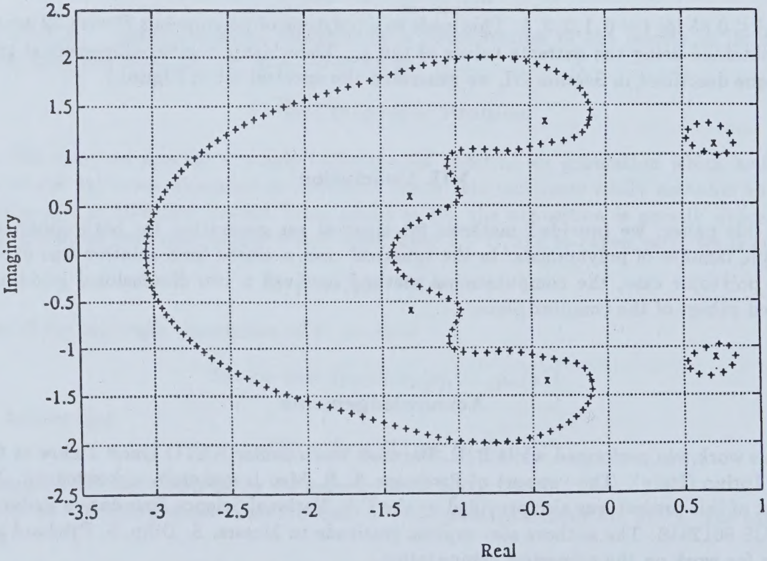


Figure 1: Spectral Set for a Spherical Family

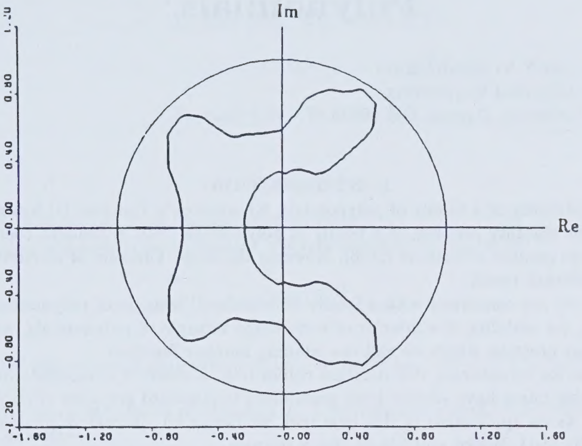


Figure 2: Spectral Set for a Polytopic (Interval Polynomial) Family

On The Stability Of Polygons Of Polynomials

L.R. PUJARA and N.R. SHANBHAG
Department of Electrical Engineering
Wright State University, Dayton, OH 45435

I. INTRODUCTION

In the area of stability of a family of polynomials, Kharitonov's Theorem [1] has generated a great deal of interest. The stability problem of a family of polynomials with dependent coefficients has been researched by a large number of authors [2]-[6]. However the Edge Theorem of Bartlett, Hollot and Lin [6] is the most significant result.

In this paper, we are concerned with a family of 'standard' triangle of polynomials (to be defined later). We explore, for stability, the interior of a standard triangle of polynomials, with the help of a frequency-dependent function which we call the winding number function.

The motivation for introducing this function comes from a result of Chapellat and Bhattacharyya [7]. The results in this paper have already been generalized to standard polygons of polynomials and will appear elsewhere. As an application of this new tool, we derive an alternative proof of the celebrated Edge Theorem of Bartlett, Hollot and Lin [6], for a triangle.

II. PRELIMINARIES

In this paper, we will deal only with a family of polynomials with positive coefficients. As is customary, an n^{th}-degree polynomial will be identified with point in R^{n+1}.

Definition : A polygon of polynomials is a convex hull of a finite number of coplanar polynomials.

Definition : A polygon of polynomials will be called standard if

(i) all the polynomials are of the same degree and have all positive coefficients.
(ii) its vertices don't have any zeros on the imaginary axis.
(iii) atmost one polynomial in the polygon vanishes at $j\omega_0$ for any $\omega_0 \geq 0$.

In this paper, we deal only with a 'standard' triangle of polynomials.

We denote $f_o(s)/s$ by $f^o(s)$, where $f_o(s)$ is the usual odd part of a polynomial $f(s)$. Also $f^e(s)$ denotes the even part of $f(s)$.

Definition :

$$X_{1,2}(\omega) = f_1^e(\omega)f_2^o(\omega) - f_2^e(\omega)f_1^o(\omega)$$
$$X_{2,3}(\omega) = f_2^e(\omega)f_3^o(\omega) - f_3^e(\omega)f_2^o(\omega)$$
$$X_{3,1}(\omega) = f_3^e(\omega)f_1^o(\omega) - f_1^e(\omega)f_3^o(\omega)$$

where $f_1(s)$, $f_2(s)$ and $f_3(s)$ represent the vertices of a standard triangle and $s = j\omega$.

Definition :

$$S(\omega) = Sgn(X_{1,2}(\omega)) + Sgn(X_{2,3}(\omega)) + Sgn(X_{3,1}(\omega))$$

will be called the winding number function for a standard triangle where

$$Sgn(X_{i,j}(\omega)) = \begin{cases} 1 & \text{if } X_{i,j}(\omega) > 0 \\ -1 & \text{if } X_{i,j}(\omega) < 0 \\ 0 & \text{if } X_{i,j}(\omega) = 0 \end{cases}$$

The following result by Chapellat and Bhattacharyya [7] is needed.

Lemma 1: *A convex combination of two polynomials , $f_1(s)$ and $f_2(s)$, vanishes at some $s = j\omega_0$ iff*

$$f_1^e(\omega_0)f_2^o(\omega_0) - f_1^o(\omega_0)f_2^e(\omega_0) = 0$$

and

$$f_1^e(\omega_0)f_2^e(\omega_0) \leq 0 \qquad and \qquad f_1^o(\omega_0)f_2^o(\omega_0) \leq 0$$

III. MAIN RESULTS

Theorem 1: *A polynomial $f(s)$ in a standard triangle vanishes at $j\omega_0$ iff*

$$X_{i,j}(\omega_0) \geq 0, \quad \forall i, \quad i = 1,2,3$$

or

$$X_{i,j}(\omega_0) \leq 0, \quad \forall i, \quad i = 1,2,3$$

and

$$S(\omega_0) \neq 0$$

To establish this theorem, we need the following lemmas.

Lemma 2 *:If a polynomial in a standard triangle vanishes at $j\omega_0$, then*

$$S(\omega_0) \neq 0$$

Lemma 3 *:If any two of $X_i(\omega)$ vanish at ω_0, for some ω_0, and $1 \leq i \leq 3$, then the third $X_i(\omega)$ also vanishes at ω_0.*

We now derive some corollaries of Theorem 1. A polynomial is said to be *within* a triangle if it is in it but not on its edges.

Cor.1 : A polynomial *within* a standard triangle vanishes at $j\omega_0$, iff $S(\omega_0) = \pm 3$.

Cor.2 : A polynomial $f(s)$ on an edge of a standard triangle vanishes at $j\omega_0$ iff $S(\omega_0) = \pm 2$.

Cor.3 : If a polynomial $f(s)$ outside a standard triangle vanishes at $j\omega_0$ then $|S(\omega_0)| < 2$.

As an application of the ideas developed so far, we can give a proof of a special case of the Edge Theorem [6], which we will call the Triangle Edge Theorem.

Theorem 2 *(The Triangle Edge Theorem) : If the edges of a triangle are stable then the interior, of the triangle, is stable.*

References.

[1] V.L. Kharitonov,"Asymptotic stability of an equilibrium position of a family of systems of linear differential equations", *Differential Uravnen.*, vol.14, pp. 2086-2088, 1978.

[2] B.R. Barmish,"A generalization of Kharitonov's four-polynomial concept for robust stability problems with linearly dependent coefficient perturbations", *IEEE Trans. Automat. Contr.*, vol. 34, pp.157-165, Feb. 1989.

[3] B.C. Chang, O. Ekdal, H.H. Yeh and S.S. Banda, "Computation of the real structured singular values via polytopic polynomials", *AIAA Guidance and Control Conference*, 1989.

[4] A. Sideris, " A polynomial time algorithm for checking the robust stability of a polytope of polynomials", *American Control Conf.*, 1989.

[5] L.R. Pujara, "On the stability of uncertain polynomials with dependent coefficients", to appear in the *IEEE Trans. Automat. Contr.* in 1990.

[6] A.C. Bartlett, C.V. Hollot and H. Lin, "Root locations of an entire polytope of polynomials: it suffices to check the edges", *Mathematics of Control, Signals and Systems*, vol. 1, pp.61-71, 1988.

[7] H. Chapellat and S.P. Bhattacharyya, " An alternative proof of Kharitonov's Theorem", *IEEE Trans. Automat. Contr.*, vol.34, pp. 448-450, April 1989.

ALGEBRAIC VS ANALYTIC DESIGN LIMITATIONS IMPOSED BY ILL-CONDITIONED PLANTS

J.S. FREUDENBERG
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2122

ABSTRACT

In this paper we distinguish the classes of tradeoffs present in linear feedback design. (The terminology "algebraic" or "analytic" is motivated by the type of mathematics used to describe each class of tradeoff; systems interpretations are given in the paper.) A long-standing conjecture has been that ill-conditioned plants are inherently difficult to robustly control. The elusiveness of a proof of this conjecture has motivated study of an alternate design problem, involving only the nominal plant, in the hopes that an inherent difficulty due to plant conditioning might emerge. We show in this paper that, under mild assumptions, such a design limitation must be *analytic*, rather than *algebraic*, in nature. This analysis is also interpreted as implying the same conclusion for the original robust performance problem.

1. INTRODUCTION AND MOTIVATION

Consider the linear time-invariant feedback system depicted in Figure 1, where P is the plant transfer function and C that of the compensator.

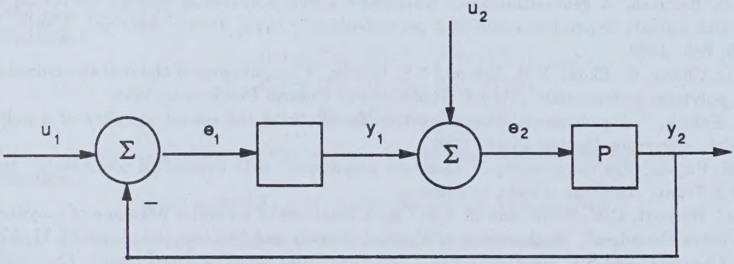


Figure 1: Feedback System

Associated with this feedback system are several important transfer functions; namely, the open loop transfer function, sensitivity function, and complementary sensitivity function defined at the plant output, $L_O = PC$, $S_O = (I + L_O)^{-1}$, and $T_O = L_O(I + L_O)^{-1}$, and their analogues defined at the plant input $L_I = CP$, $S_I = (I + L_I)^{-1}$, and $T_I = L_I(I + L_I)^{-1}$. It is well-known (e.g. [DoS81], [SLH81]) that many design

---

Research supported in part by AFOSR Contract No. F33615-88-C-3601 and in part by NSF Grant ECS-8857510.

specifications may be stated in the form of frequency-dependent *bounds* upon the closed loop transfer functions $S_O, S_I, T_O, T_I, S_O P$, and $P^{-1}T_O$. In particular, each of these six transfer functions expresses the response of a signal e_i or y_i to an exogenous input u_i (e.g. [Vid85]). It is also known that a given set of such design specifications may not be achievable, due to inherent performance limitations and tradeoffs that arise in feedback design.

Design tradeoffs and limitations may be divided into two categories, which we shall term *algebraic* and *analytic*, after the type of mathematics used to quantify their effects. Algebraic (also known as "constant matrix") tradeoffs take place between system properties at a fixed frequency, and are quantified by evaluating a transfer function identity at that frequency, resulting in an algebraic equation. For example, consider the well-known tradeoff between those feedback properties governed by the output sensitivity and complementary sensitivity functions, and quantified by the identity $S_O(j\omega) + T_O(j\omega) = I$. These tradeoffs may be viewed as a consequence of the feedback structure. Analytic tradeoffs, on the other hand, take place between system properties in *different* frequency ranges, and may be viewed as a consequence of stability or causality. They arise because the transfer function of a stable causal linear system is a complex analytic function of the frequency variable "s", and hence must satisfy Cauchy's integral theorem. Examples are the tradeoffs described by the Bode gain-phase relation, the Bode sensitivity integral, and the Poisson integral.

It has been a long-standing conjecture that ill-conditioned plants are inherently difficult to control, and impose significant limitations upon the class of achievable design specifications ([Ste84], [Ste85], [Ste87], [FrL86], [Fre89a,b], [NeM87], [SkM86]). By an ill-conditioned plant, we mean one whose transfer function matrix, evaluated at a frequency of interest, has a large condition number. Physically, this means that there exists a large directional variation in the gain of the plant when responding to signals at that frequency.

The conjecture that ill-conditioned plants impose design limitations seems first to have arisen in the study of the robust performance problem, wherein a bound upon the output sensitivity function must remain satisfied despite the presence of multiplicative uncertainty at the plant input. It has been shown that an injudicious choice of compensator can yield a design which is extremely sensitive to small modelling errors; however, a definitive analysis of the design limitations has been hard to accomplish. The difficulty in studying the robust performance problem has motivated Nett [Net89] to study an alternate design problem involving only the *nominal* system. The idea is that by requiring bounds upon several closed loop transfer functions, an inherent limitation due to plant ill-conditioning might arise. The analysis in [Net89] is only partial, however.

As a step towards understanding whether ill-conditioned plants are indeed difficult to control, it is of considerable interest to know whether the design limitations, if they indeed exist, are of the algebraic or the analytic type. This knowledge would, at a minimum, serve to steer research away from fruitless paths of inquiry, and might suggest useful research directions. Our goal in this paper is to show that, under reasonable assumptions, ill-conditioned plants do not pose any inherent *algebraic* design limitations. As a consequence, it is necessary to use the methods of complex variable theory to study the ill-conditoned plant problem.

The remainder of this paper is organized as follows. In Section 2, we study the scalar version of the design problem posed by Nett, and derive a set of necessary and sufficient conditions for a set of design specifications to be achievable at a fixed frequency. Although this analysis is straightforward, to the author's knowledge it has not heretofore appeared in the literature. Section 3 is devoted to the multivariable version of Nett's problem, and we show that the analysis of the multivariable problem may be reduced to that of a set of scalar problems. In Section 4 we show that the analysis of Section 3 may be interpreted to conclude that ill-conditioned plants pose no inherent algebraic design limitations for the original robust performance problem. Proofs are omitted due to space limitations, but are included in the full version of this report [Fre89d].

2. THE SCALAR CASE

For a scalar system, the two sets of transfer functions defined at the plant input and output are identical, and we therefore omit the subscript "I" or "O" in this section. We shall consider a design specification requiring

$$|r_1(j\omega)T(j\omega)| \leq 1 \tag{2.1}$$

$$|r_2(j\omega)S(j\omega)| \leq 1 \tag{2.2}$$

$$|r_3(j\omega)P^{-1}(j\omega)T(j\omega)| \leq 1 \tag{2.3}$$

$$|r_4(j\omega)S(j\omega)P(j\omega)| \leq 1 \ , \tag{2.4}$$

where $r_1 - r_4$ are weighting functions used to reflect the relative importance of system properties at different frequencies. Our task is to determine, for a given set of weightings, whether the identity

$$S(j\omega) + T(j\omega) = 1 \tag{2.5}$$

prevents these specifications from being simultaneously satisfied at a given frequency. For example, if a specification (2.2) or (2.4) upon $|S(j\omega)|$ is satisfied, the identify (2.5) imposes a constraint upon $|T(j\omega)|$, and thus upon the ability of (2.1) or (2.3) to be achieved.

For most of this section, we shall work only with system properties at a fixed frequency, and shall therefore suppress dependence upon the frequency variable.

Lemma 2.1: Consider the design specifications (2.1-4) evaluated at a fixed frequency.

(a) If (2.1) is satisfied, then

$$|r_2S| \ \geq \ |r_2|\,[1 - 1/|r_1|] \tag{2.6}$$

$$|r_4SP| \ \geq \ |r_4|\ |P|\,[1 - 1/|r_1|] \tag{2.7}$$

(b) If (2.2) is satisfied, then

$$|r_1T| \ \geq \ |r_1|\,[1 - 1/|r_2|] \tag{2.8}$$

$$|r_3P^{-1}T| \ \geq \ |r_3|\ |P^{-1}|\,[1 - 1/|r_2|] \tag{2.9}$$

(c) If (2.3) is satisfied, then

$$|r_2S| \geq |r_2|\left[1 - 1/|r_3||P^{-1}|\right] \tag{2.10}$$
$$|r_4SP| \geq |r_4||P|\left[1 - 1/|r_3|\,|P^{-1}|\right] \tag{2.11}$$

(d) If (2.4) is satisfied, then

$$|r_1T| \geq |r_1|\,[1 - 1/|r_4||P|] \tag{2.12}$$
$$|r_3P^{-1}T| \geq |r_3||P^{-1}|\,[1 - 1/|r_4||P|] \tag{2.13}$$

■

The following proposition characterizes, in terms of the weightings and the plant gain, those specifications that are simultaneously achievable. Notice that if these specifications are indeed achievable, then the proof technique gives a constructive procedure for finding a "design" which satisfies the specifications at the given frequency.

Proposition 2.2: Assume that $P \neq 0$ and $|P| \neq \infty$. Then the specifications (2.1)-(2.4) can be satisfied simultaneously at a given frequency if and only if the following bounds are satisfied

$$\frac{1}{|r_1|} + \frac{1}{|r_2|} \geq 1 \tag{2.14}$$
$$\frac{1}{|r_1|} + \frac{1}{|r_4P|} \geq 1 \tag{2.15}$$
$$\frac{1}{|r_3P^{-1}|} + \frac{1}{|r_2|} \geq 1 \tag{2.16}$$
$$\frac{1}{|r_3P^{-1}|} + \frac{1}{|r_4P|} \geq 1 \tag{2.17}$$

■

The sufficiency part of the above proof shows that if we choose weightings so that (2.14-17) are satisfied at *all* frequencies, then any obstacle preventing (2.1-4) from being satisfied is *necessarily* due to frequency dependence and thus cannot be analyzed merely by studying the algebraic constraints at each frequency imposed by (2.14-17).

The bound (2.14) is a consequence of the tradeoff between making $|S|$ and $|T|$ simultaneously small; i.e., for (2.1) and (2.2) to be simultaneously achievable, $|r_1|$ and $|r_2|$ cannot both be large. Typically, this tradeoff is resolved by requiring these weightings to have the form shown in Figure 2. Following Stein [Ste87], we shall term the intermediate frequency range $[\omega_1, \omega_2]$ the *crossover gap*. Notice that weightings of this form satisfy the more general condition

$$\min\{|r_1|,\ |r_2|\} < 1 \tag{2.18}$$

at all frequencies, and thus (2.14) is trivially satisfied.

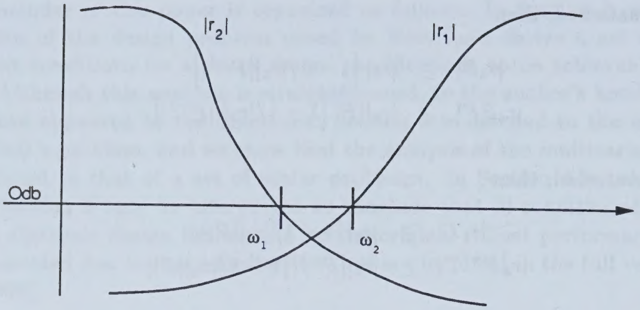


Figure 2: Crossover Gap

The bounds (2.15-17) are also consequences of the tradeoff between $|S|$ and $|T|$; however, in each case the severity of the tradeoff depends upon the value of the plant gain. It is instructive to assume that r_1 and r_2 are given, and then to describe those weightings r_3 and r_4 for which these bounds are satisfied.

Lemma 2.3: Suppose that weightings r_1 and r_2 are as depicted in Figure 2. Then the bound (2.14) is trivially satisfied at all frequencies, and conditions *sufficient* for the bounds (2.15-17) to be satisfied are that

(a) For $\omega < \omega_1$, $|r_3| < |P|$ (2.19)

(b) For $\omega > \omega_2$, $|r_4| < |P^{-1}|$ (2.20)

(c) For $\omega \in (\omega_1, \omega_2)$, $|r_3| < |P|$ or
$|r_4| < |P^{-1}|$ (2.21)

■

Alternately, one can consider a fixed set of weightings $r_1 - r_4$ and characterize those values of the plant gain for which (2.15 - 2.17) are satisfied. In this regard it is interesting to note that (2.17) will *always* hold for plant gains that are very large or very small. Indeed, one can calculate the worst case value of the plant gain, and use this value to derive a constraint upon r_3 and r_4 which guarantees that (2.17) will be satisfied *independently* of the plant gain.

Lemma 2.4 Consider the function

$$F(\alpha) \triangleq \frac{\alpha}{|r_3|} + \frac{1}{|r_4|\alpha} \quad . \tag{2.22}$$

The minimum value of $F(\alpha)$ occurs at

$$\alpha^* = \sqrt{|r_3|/|r_4|} \tag{2.23}$$

and is given by

$$F(\alpha^*) = 2/\sqrt{|r_3||r_4|} \tag{2.24}$$

It follows that if we choose r_3 and r_4 so that

$$|r_3||r_4| \leq 4 \quad , \tag{2.25}$$

then (2.17) is trivially satisfied. For example, one might impose this condition in the crossover gap as an alternative to (2.21). These observations will prove useful in our study of multivariable systems.

3. THE MULTIVARIABLE CASE

In this section we develop multivariable extensions of the results in Section 2. We show that the problem of determining whether a set of design specifications is simultaneously achievable (at a fixed frequency) for a multivariable plant reduces to that of determining whether those specifications are simultaneously achievable for a set of scalar plants. The role of gain for each plant in this set is played by a singular value of the plant transfer function. We shall argue that, under mild assumptions, ill-conditioned plants pose no inherent design limitations that may be analyzed using *algebraic* equations.

One difference between the scalar and the multivariable case is that, in general, the transfer functions defined at the plant input will differ from those at the plant output. Hence, in principle, we need to consider specifications upon *all six* transfer functions $S_O, S_I, T_O, T_I, S_O P$, and $P^{-1}T_O$. As it turns out, there is no loss of generality in assuming that the specifications imposed upon S_O and S_I (similarly, upon T_O and T_I) are *identical.* We shall return to this point after deriving our main results. As in the previous section, we work primarily with system properties at a fixed frequency, and therefore suppress dependence upon the frequency variable when convenient.

Our design specifications will require that

$$\bar{\sigma}[r_1(j\omega)T_I(j\omega)] \leq 1 \tag{3.1}$$

$$\bar{\sigma}[r_2(j\omega)S_O(j\omega)] \leq 1 \tag{3.2}$$

$$\bar{\sigma}[r_3(j\omega)P^{-1}(j\omega)T_O(j\omega)] \leq 1 \tag{3.3}$$

$$\bar{\sigma}[r_4(j\omega)S_O(j\omega)P(j\omega)] \leq 1 \quad . \tag{3.4}$$

The algebraic identities imposing potential design tradeoffs are

$$S_O + T_O = I \tag{3.5}$$

$$S_I + T_I = I \quad . \tag{3.6}$$

Also needed are the identities

$$PS_O = S_I P \tag{3.7}$$

$$P^{-1}T_O = T_I P^{-1} \quad . \tag{3.8}$$

Let the singular values of the plant transfer function be denoted $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_n$. It will sometimes be convenient to use the alternate notation $\bar{\sigma}[P] = \tau_1$ and $\bar{\sigma}[P^{-1}] =$

$1/\tau_n$. The following results are multivariable analogues to Lemma 2.1 and Proposition 2.2.

Lemma 3.1: Consider the design specifications (3.1-4) evaluated at a fixed frequency.

(a) If (3.1) is satisfied, then

$$\bar{\sigma}[r_2 S_O] \geq |r_2|\,[1 - 1/|r_1|] \tag{3.9}$$

$$\bar{\sigma}[r_4 S_O P] \geq |r_4|\bar{\sigma}[P]\,[1 - 1/|r_1|] \tag{3.10}$$

(b) If (3.2) is satisfied, then

$$\bar{\sigma}[r_1 T_I] \geq |r_1|\,[1 - 1/|r_2|] \tag{3.11}$$

$$\bar{\sigma}[r_3 P^{-1} T_O] \geq |r_3|\bar{\sigma}[P^{-1}]\,[1 - 1/|r_2|] \tag{3.12}$$

(c) If (3.3) is satisfied, then

$$\bar{\sigma}[r_2 S_O] \geq |r_2|\left[1 - 1/|r_3|\bar{\sigma}[P^{-1}]\right] \tag{3.13}$$

$$\bar{\sigma}[r_4 S_O P] \geq |r_4|\tau_i\,[1 - \tau_i/|r_3|] \quad \forall i \tag{3.14}$$

(d) If (3.4) is satisfied, then

$$\bar{\sigma}[r_1 T_I] \geq |r_1|\,[1 - 1/|r_4|\bar{\sigma}[P]] \tag{3.15}$$

$$\bar{\sigma}[r_3 P^{-1} T_O] \geq |r_3|(1/\tau_i)\,[1 - 1/|r_4|\tau_i]\,, \quad \forall i \tag{3.16}$$

■

Proposition 3.2: The bounds (3.1)-(3.4) may be achieved simultaneously at any frequency for which

$$1/|r_1| + 1/|r_2| \geq 1 \tag{3.17}$$

$$1/|r_2| + \underline{\sigma}[P]/|r_3| \geq 1 \tag{3.18}$$

$$1/|r_4|\bar{\sigma}[P] + 1/|r_1| \geq 1 \tag{3.19}$$

$$1/|r_4|\tau_i + \tau_i/|r_3| \geq 1\,, \quad \forall i \tag{3.20}$$

■

It is easy to verify that, if bounds were also imposed upon T_O and S_I, inequalities similar to (3.17-20) could be derived, with the bounds on T_O and S_I replacing those upon T_I and S_O, respectively. Hence there is no loss of generality in assuming that the two sensitivity and the two complementary sensitivity functions must satisfy the same design specifications.

From Proposition 3.2 it is clear that the multivariable problem reduces to a set of n scalar problems. Specifically, we must choose weightings $r_1 - r_4$ such that each of the n scalar problems obtained by setting $|P| = \tau_i$ in Proposition 2.2 yields an achievable design specification. The following is an analogue to Lemma 2.3.

Lemma 3.3: Suppose that weightings r_1 and r_2 are as depicted in Figure 2. Then the bound (3.17) is trivially satisfied at all frequencies and conditions *sufficient* for the bounds (3.18-20) to be satisfied are that

(a) For $\omega < \omega_1$, $|r_3| < \underline{\sigma}[P]$ (3.21)

(b) For $\omega > \omega_2$, $|r_4| < \underline{\sigma}[P^{-1}]$ (3.22)

(c) For $\omega \in (\omega_1, \omega_2)$ $|r_3| < \tau_i$ or $|r_4| < 1/\tau_i$, $\forall i$. (3.23)

■

As an alternative to (c), one could require that (2.25) hold.

Recall that the plant condition number is defined, at each frequency, by $\kappa[P] = \bar{\sigma}[P]/\underline{\sigma}[P]$, and may be interpretated as the ratio of largest to smallest gains at that frequency. A plant with a large condition number is said to be *ill-conditioned*, and the conjecture referred to in Section 1 is that ill-conditioned plants are inherently difficult to control. In the present context, we wish to know whether our ability to satisfy the specifications (3.1-4) decreases as the plant condition number increases. We study this problem by studying the bounds (3.17-20).

Clearly, our ability to satisfy (3.17) does not depend upon the plant condition number. Furthermore, since (3.20) is always satisfied for sufficiently large and/or small values of gain, a large condition number poses no inherent difficulty in satisfying this bound. This leaves the bounds (3.19-20). Achievability of (3.19) is an issue only if $|r_2| > 1$. In that case the bound will always be violated for a sufficiently small value of $\underline{\sigma}[P]$. This, however, is not a function of the condition number *per se* as $\kappa[P]$ increasing does not necessarily imply that $\underline{\sigma}[P]$ decreases. Similarly, achievability of (3.20) is an issue only if $|r_1| > 1$, and the bound will always be violated for a sufficiently large value of $\bar{\sigma}[P]$. Again, this is not directly a function of the plant condition number as $\kappa[P]$ increasing need not imply that $\bar{\sigma}[P]$ increases. Indeed, only if both $|r_1| > 1$ and $|r_2| > 1$ at the same frequency does the condition number increasing imply that either (3.19) or (3.20) will be violated. In that case, these bounds are simultaneously achievable only if the condition number satisfies the following bound. (This bound was derived, under overly loose assumptions, in [Net89].)

Corollary 3.4: Suppose that the bounds (3.1-4) are simultaneously satisfied. Suppose also that $|r_1| > 1$ and/or $|r_2| > 1$. Then, necessarily,

$$1 \geq \kappa[P]|r_3|\,|r_4|\,[1 - 1/|r_1|]\,[1 - 1/|r_2|] \quad . \tag{3.24}$$

■

Obviously, this bound is vacuous unless both $|r_1| > 1$ and $|r_2| > 1$ at the same frequency. Since this is inconsistent with the presence of a crossover gap in Figure 2 (more generally, with (2.18), and since we claim that the existence of a crossover gap is a very plausible assumption, it follows that an ill-conditioned plant poses no inherent *algebraic* design limitation.

4. THE ROBUST PERFORMANCE PROBLEM

The conjecture that ill-conditioned plants are inherently difficult to control appears to have first arisen in the study of the robust performance problem of maintaining the output sensitivity function small despite unstructured multiplicative uncertainty at the plant input. For the present purpose, it suffices to note that robust performance is present if and only if the *structured singular value* satisfies

$$\mu[M(j\omega)] < 1 \quad , \quad \forall\omega \quad , \tag{4.1}$$

where

$$M = \begin{bmatrix} r_1 T_I & -r_1 P^{-1} T_O \\ -r_2 S_O P & r_2 S_O \end{bmatrix} \tag{4.2}$$

Here r_1 and r_2 are weighting functions having the general shape shown in Figure 2. (See [Fre89a,b] for details.)

Lemma 5.1: At each frequency

$$\begin{aligned} \mu[M] &\leq \max\{\bar{\sigma}[r_1 T_I],\ \bar{\sigma}[r_2 S_O]\} \\ &+ \sqrt{\bar{\sigma}[r_1 P^{-1} T_O]\bar{\sigma}[r_2 S_O P]} \end{aligned} \tag{4.3}$$

$$\begin{aligned} \mu[M] \geq \max\Big\{ & \bar{\sigma}[r_1 T_I],\ \bar{\sigma}[r_2 S_O] \ , \\ & \sqrt{\bar{\sigma}[r_1 P^{-1} T_O]\bar{\sigma}[r_2 S_O P]} \Big\} \end{aligned} \tag{4.4}$$

The upper bound (4.3) appears in [Fre89a], while the lower bound (4.4), due to Nett [Net89], is a tightening of one also appearing in [Fre89a]. Using these bounds, it is clear that the problem of satisfying (4.1) is approximately equivalent to requiring that

$$\bar{\sigma}[r_1 T_I] < 1 \tag{4.5}$$

$$\bar{\sigma}[r_2 S_O] < 1 \tag{4.6}$$

$$\sqrt{\bar{\sigma}[r_1 P^{-1} T_O]\bar{\sigma}[r_2 S_O P]} < 1 \ . \tag{4.7}$$

(Alternately, one could obtain a sufficient condition by requiring each to be less than 1/2.)

The point is that (4.5-7) are *less* stringent requirements than are (3.1-4). Hence, under the reasonable assumption of a crossover gap, an ill-conditioned plant poses no inherent algebraic design limitation for the robust performance problem.

REFERENCES

[DoS81] Doyle, J.C., and Stein, G., "Multivariable feedback design: concepts for a classical/modern synthesis," *IEEE Trans. on Auto. Contr.*, vol. AC-26, Feb. 1981.

[FrL86] Freudenberg, J.S., and Looze, D.P., "Relations between properties of multivariable feedback systems at different loop-breaking points: part II," *Proc. 1986 American Contr. Conf.*, Seattle, WA, June 1986.

[Fre89a] J.S. Freudenberg, "Analysis and Design to Ill-Conditioned Plants Part I: Lower Bounds on the Structured Singular Value", *Int. J. Control*, vol. 49, 1989.

[Fre89b] J.S. Freudenberg, "Analysis and Design of Ill-Conditioned Plants Part II: Directionally Uniform Weightings and an Example", *Int. J. Control*, vol. 49, 1989.

[Fre89d] J.S. Freudenberg, "Algebraic vs Analytic Design Limitations Imposed by Ill-Conditioned Plants", Technical Report, University of Michigan, 1989.

[NeM87] Nett, C.N. and Manousiouthakis, V., "Euclidean condition and block relative gain: connections, conjectures, and clarifications", *IEEE Trans. Auto. Contr.*, vol. AC-32, No. 5, pp. 405-407, May 1987.

[Net89] C. Nett, personal communication, 1989.

[SLH81] Safonov, M.G., Laub, A.J. and Hartmann, G.L., "Feedback properties of multivariable systems: the role and use of the return difference matrix," *IEEE Trans. on Auto. Contr.*, vol. AC-26, 1981.

[SkM86] Skogestad, S. and Morari, M., "Control of ill-conditioned plants: high purity distillation", *AICHE annual Meeting*, Nov. 1986.

[Ste73] Stewart, G.W., *Introduction to Matrix Computations*, NY: Academic Press, 1973.

[Ste84] Stein, G., *Notes*, ONR/Honeywell Workshop, Oct. 1984.

[Ste85] Stein, G., "Beyond singular values and loop shapes," Report No. LIDS-P-1504, MIT., Cambridge, MA, Aug. 1985.

[Ste87] Stein, G., *Lecture Notes of Workshop on Robust Multivariable Control*, Minneapolis, MN, June 1987.

NEAR-OPTIMUM STEADY STATE REGULATORS FOR STOCHASTIC LINEAR WEAKLY COUPLED DISCRETE SYSTEMS

XUEMIN SHEN and ZORAN GAJIC

Rutgers University, Department of Electrical and
Computer Engineering, Piscataway, NJ, 08855-0909.

The linear weakly coupled systems were introduced in control theory [1]. The linear quadratic Gaussian control problem of weakly coupled discrete systems has not been studied in the literature yet. This is due to the fact that the partitioned form of the main equation of the optimal linear control theory - Riccati equation, has a very complicated form in the discrete time domain. In [2] that problem is overcome by the use of the bilinear transformation [5], which is applicable under quite mild assumption. The weakly coupled structure of the global Kalman filter is exploited in this paper such that it may be replaced by two lower order local filters. This has been achieved via the use of the decoupling transformation introduced in [3].

The paper presents the approach to the decomposition and approximation of the linear quadratic Gaussian (LQG) control problem of weakly coupled discrete systems. The decoupling transformation of [3] is used for the exact block diagonalization of the global Kalman filter. The approximate feedback control law is then obtained by approximating the coefficients of the optimal local filters with the accuracy of $O(\varepsilon^N)$. It is shown that the order of approximation of the optimal performance is $O(\varepsilon^N)$, and the order of approximation of the optimal system trajectories is $O(\varepsilon^{2N})$. As a result, under very mild conditions, we have achieved the reduction in both off-line and on-line computational requirements.

MAIN RESULT

Consider a linear weakly coupled discrete stochastic system:

$$x_1(n+1)=A_{11}x_1(n)+\varepsilon A_{12}x_2(n)+B_{11}u_1(n)+\varepsilon B_{12}u_2(n)+G_{11}w_1(n)+\varepsilon G_{12}w_2(n) \qquad (1a)$$

$$x_2(n+1)=\varepsilon A_{21}x_1(n)+A_{22}x_2(n)+\varepsilon B_{21}u_1(n)+B_{22}u_2(n)+\varepsilon G_{21}w_1(n)+G_{22}w_2(n) \qquad (1b)$$

$$y_1(n)=C_{11}x_1(n)+\varepsilon C_{12}x_2(n)+v_1(n)\ , \quad y_2(n)=\varepsilon C_{21}x_1(n)+C_{22}x_2(n)+v_2(n) \qquad (2)$$

with a performance criterion

$$J = E\{\sum_{n=0}^{\infty}[z^T(n)z(n) + u_1^T(n)R_1u_1(n) + u_2^T(n)R_2u_2(n)]\} \qquad (3)$$

where $x_i\epsilon\mathbf{R}^{n_i}$ comprise state vectors, $u_i\epsilon\mathbf{R}^{m_i}$ are control inputs, $y_i\epsilon\mathbf{R}^{l_i}$ are observed outputs, $w_i\epsilon\mathbf{R}^{r_i}$ and $v_i\epsilon\mathbf{R}^{l_i}$ are independent zero-mean stationary white Gaussian noise mutually uncorrelated processes with intensities $W_i>0$ and $V_i>0$, respectively, R_i's are positive definite matrices and $z_i\epsilon\mathbf{R}^{s_i}$, i=1,2, are controlled outputs given by:

$$z_1(n)=D_{11}x_1(n)+\varepsilon D_{12}x_2(n), \quad z_2(n)=\varepsilon D_{21}x_1(n)+D_{22}x_2(n) \tag{4}$$

All matrices are bounded functions of a small coupling parameter ε, having appropriate dimensions.

The optimal control law is given by a very well known form:

$$u(n) = - F\hat{x}(n) \tag{5}$$

with

$$\hat{x}(n+1) = A\hat{x}(n) + B(n) + K[y(n) - C\hat{x}(n)] \tag{6}$$

where K and F are the filter gain and regulator gain.

The proposed approximate control laws are defined:

$$u_1^{(k)}(n)=-f_{11}^{(k)}\hat{\eta}_1^{(k)}(n)-\varepsilon f_{12}^{(k)}\hat{\eta}_2^{(k)}(n), \quad u_2^{(k)}(n)=-\varepsilon f_{21}^{(k)}\hat{\eta}_1^{(k)}(n)-f_{22}^{(k)}\hat{\eta}_2^{(k)}(n) \tag{7}$$

with

$$\hat{\eta}_1^{(k)}(n+1) = \alpha_1^{(k)}\hat{\eta}_1^{(k)}(n) + \beta_{11}^{(k)}\nu_1^{(k)}(n) + \varepsilon\beta_{12}^{(k)}\nu_2^{(k)}(n)$$

$$\hat{\eta}_2^{(k)}(n+1) = \alpha_2^{(k)}\hat{\eta}_2^{(k)}(n) + \varepsilon\beta_{21}^{(k)}\nu_1^{(k)}(n) + \beta_{22}^{(k)}\nu_2^{(k)}(n) \tag{8}$$

where $\hat{\eta}_1$ and $\hat{\eta}_2$ are completely decoupled reduced-order filters; ν_1 and ν_2 are innovation processes. All coefficients are defined in [4].

The near-optimality of the proposed approximate control law (7) is established in the following theorem.

Theorem: Let x_1 and x_2 be the optimal trajectories and J be the optimal value of the performance criterion. Let $x_1^{(k)}$, $x_2^{(k)}$ and $J^{(K)}$ be the corresponding quantities under the approximate control law $u^{(k)}$ given by (7). Then the following hold

$$J^{opt} - J^{(k)} = O(\varepsilon^k), \quad Var\{ x_i - x_i^{(k)} \} = O(\varepsilon^{2k}) \quad k = 0,1,2,\ldots . \tag{9}$$

The proof of this theorem can be found in [4].

REFERENCES

[1] P. Kokotovic, W. Perkins, J. Cruz Jr. and D'Ans, Proceeding IEE, Vol.116, 889-992, 1969.

[2] X. Shen and Z. Gajic, "Optimal Reduced Order Solution of the Weakly Coupled Discrete Systems", submitted for publication.

[3] Z. Gajic and X. Shen, Int. J. Control, 1989.

[4] X. Shen, Ph.D Dissertation, Rutgers University, 1989.

[5] R. Kondo and K. Furuta, IEEE Trans. Automatic Control, AC-31, 50-54, 1986.

NONLINEAR OUTPUT REGULATION: REMARKS ON ROBUSTNESS

C.I. BYRNES AND A. ISIDORI
Department of Systems Science and Mathematics
Washington University
St. Louis, MO 63130

ABSTRACT

Necessary and sufficient conditions are given for the solution of a general nonlinear output regulation problem. Straightforward application of these conditions shows that, somehow surprisingly, a "robust" linear regulator cannot tolerate nonlinear perturbations in the plant parameters.

1. INTRODUCTION

One of the most important problems in linear multivariable control theory is that of controlling a fixed plant in order to have its output tracking (or rejecting) reference (or disturbance) signals produced by some external generator (the exosystem). This problem has been treated by several authors, among whom we mention - for instance - Davison [4], Francis and Wonham [7], Francis [6]. In particular, the work of Francis has shown that the solvability of a multivariable linear regulator problem corresponds to the solvability of a system of two linear matrix equations. An ever-present concern in the synthesis of linear regulators is the so-called "robustness" issue, namely the achievement output regulation in the presence of unknown plant parameter variations (see, e.g. [5], [7]).

More recently, several authors have considered the corresponding problem for a nonlinear plant [1],[8], [9], [2]. In particular, in the paper [9], we showed how the results established by Francis can be extended to the case in which the exosystem generates time-varying reference signals and/or disturbances (including the very important case of periodic signals) and the plant is modeled by a differential equation whose right-hand-side is affine in the inputs. The further extension of these results to the case of a plant modeled by a general nonlinear differential equation in [2].

The purpose of this paper is to review the basis results of [2] and to prove that even though the solution of an output regulation problem involves exponential stability and exponential rate of decay of tracking error, the problem itself cannot be solved - in general - on the basis of the knowledge of an approximated linear model. This somehow contradicts the conjecture that robust linear regulators could tolerate any sort of perturbation.

2. NONLINEAR REGULATOR THEORY

Consider a multivariable nonlinear system modeled by differential equations of the form:

$$\dot{x} = f(x,w,u) \tag{2.1.a}$$

$$\dot{w} = s(w) \tag{2.1.b}$$

$$e = h(x,w) \tag{2.2}$$

The first equation of this system describes a *plant* with state x, defined on a neighborhood X of the origin or R^n, and input $u \in R^m$, subject to the effect of a *disturbance* and/or reference signal w. The second equation describes an autonomous system, the so-called *exosystem*, defined in a neighborhood W of the origin of R^s, which models the class of disturbance and reference signals taken into consideration. The third equation defines an *error* $e \in R^p$, expressed as a function of the state plant x and of the exosystem state w.

The mappings $f: X \times W \times R^m \to R^n$, $s: W \to R^s$, $h: X \times W \to R^p$ are smooth mappings. We assume also that $f(0,0,0)=0$, $s(0)=0$, $h(0,0)=0$. Thus, for $u=0$, the composite system (2.1) has an equilibrium state $(x,w)=(0,0)$ with zero error (2.2).

The control action to (2.1) is provided by an *error feedback controller*, i.e. a dynamical system

$$u = \theta(z) \tag{2.3a}$$

$$\dot{z} = \eta(z,e) \tag{2.3b}$$

with state z, defined on a neighborhood Z of the origin of R^v. For each $e \in R^p$, $\eta(z,e)$ is a C^k vector field on Z, and $\theta(z)$ is a C^k mapping defined on Z (for some integer $k \geq 2$). The composition of (2.1) with (2.3) yields a closed loop system:

$$\dot{x} = f(x,w,\theta(z)) \tag{2.4a}$$

$$\dot{z} = \eta(z,h(x,w)) \tag{2.4b}$$

$$\dot{w} = s(w) \tag{2.4c}$$

We assume $\eta(0,0)=0$ and $\theta(0)=0$, so that the triplet $(x,z,w) = (0,0,0)$ is an equilibrium of the closed loop (2.4).

The purpose of control action is to achieve local exponential stability and output regulation. *Local exponential stability* means that, *when the exosystem is disconnected* (i.e. when w is set equal to 0), the closed loop system (2.4a-2.4b) has an exponentially stable equilibrium at $(x,z)=(0,0)$. *Output regulation* means that in the closed loop system (2.6), for all initial states $(x(0),z(0),w(0))$ sufficiently close to the origin, $e(t)\to 0$ as $t\to\infty$.

Formally, all of this can be summarized in the following synthesis problem.

Error Feedback Regulator Problem

Find, if possible, $\theta(z)$ and $\eta(z,e)$, such that:

(S) the equilibrium $(x,z)=(0,0)$ of:

$$\dot{x} = f(x,0,\theta(z))$$

$$\dot{z} = \eta(z,h(x,0))$$

is exponentially stable,

(R) there exists a neighborhood $U\subset X\times Z\times W$ of $(0,0,0)$ such that, for each initial condition $(x(0),z(0),w(0))\in U$, the solution of (2.4) satisfies:

$$\lim_{t\to\infty} h(x(t),w(t)) = 0$$

The theory presented in this paper is based on a suitable set of standing assumptions that, to some extent, generalize the assumptions usually accepted in the corresponding theory of output regulation in linear systems. For convenience, we recall and/or introduce, some terminology.

Definition 1. Let $\Phi_t^f(x^0)$ denote the flow of the vector field f. A point x is called an ω-limit point of f if there exist a point x^0 and a sequence $t_n\in R$ such that $t_n\to\infty$ and $\Phi_{t_n}^f(x^0)\to x$ as $n\to\infty$. The set of all ω-limit points of f is called the ω-limit (or positive limit) set of f.

Definition 2. A system $\dot{x}=f(x,u)$, with $f(0,0)=0$, is said to be *asymptotically* (exponentially) *stabilizable* if there exists a smooth function $u=k(x)$ (the stabilizer), with $k(0)=0$, such that the equilibrium $x=0$ of $\dot{x}=f(x,k(x))$ is asymptotically (exponentially) stable.

Definition 3. A system $\dot{x}=f(x,u)$ and $y=h(x)$ with $f(0,0)=0$ and $h(0)=0$, is said to be *exponentially detectable* if there exists a system $\dot{z}=\gamma(z,y)$ (the *detector*), with $\gamma(0,0)=0$, such that

(i) $h(z,h(z)) = f(z,0)$

(ii) the equilibrium $z=0$ of $\dot{z}=\gamma(z,0)$ is exponentially stable.

We are now in a position to state the three basic assumptions on which the regulator theory is based.

(H1) The exosystem has the following properties:

(i) the equilibrium point $w=0$ is stable

(ii) the positive limit set of s is dense in W

(H2) The system

$$\dot{x} = f(x,0,u)$$

is exponentially stabilizable

(H3) The system

$$\begin{pmatrix} \dot{x} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} f(x,w,u) \\ s(w) \end{pmatrix}$$

$$e = h(x,w)$$

is exponentially detectable.

Note that (H1) implies that the Jacobian matrix $\left[\frac{\partial s}{\partial w}\right]_{w=0}$ has all eigenvalues on the imaginary axis.

Lemma 1. Suppose (H1) holds and assume that, for some $\theta(z)$ and $\eta(z,e)$ the condition (S) is fulfilled. Then, the condition (R) is also fulfilled if and only if there exist $C^k (k\geq 2)$ mappings $x=\pi(w)$, with $\pi(0)=0$, and $z=\sigma(w)$, with $\sigma(0)=0$, both defined in a neighborhood $W^o \subset W$ of 0, satisfying the conditions:

$$\frac{\partial \pi}{\partial w} s(w) = f(\pi(w),w,\theta(\sigma(w))) \tag{2.5a}$$

$$\frac{\partial \sigma}{\partial w} s(w) = \eta(\sigma(w),0) \tag{2.5b}$$

$$h(\pi(w),w) = 0 \tag{2.5c}$$

Proof. The Jacobian matrix of the right-hand-side of (2.4) at the equilibrium point $(x,z,w)=0$ has a block triangular form

$$\begin{bmatrix} \tilde{A} & * \\ 0 & S \end{bmatrix}$$

with

$$\tilde{A} = \begin{bmatrix} \dfrac{\partial f(x,w,\theta(z))}{\partial x} & \dfrac{\partial f(x,w,\theta(z))}{\partial z} \\ \dfrac{\partial \eta(z,h(x,w))}{\partial x} & \dfrac{\partial \eta(z,h(x,w))}{\partial z} \end{bmatrix}_{x=0,z=0,w=0}$$

$$S = \left[\frac{\partial s}{\partial w}\right]_{w=0}$$

By (H1), S has all eigenvalues on the imaginary axis while, by (S), $\tilde{A}$ has all eigenvalues in the open left-half plane. Thus, system in question has a center manifold at (0,0), the graph of a C^k mapping $x=\pi(w)$, $z=\sigma(w)$, with $\pi(w)$ satisfying (2.5a) and $\sigma(w)$ satisfying

$$\frac{\partial\sigma}{\partial w} s(w) = \eta(\sigma(w),h(\pi(w),w))$$

This manifold is locally invariant under the flow of (2.4) and the restriction of the flow of (2.4) to this manifold is a diffeomorphic copy of the flow of the exosystem. If w^0 is a point in the positive limit set of s, then there exists $w\in W$ such that the trajectory $(x(t),z(t),w(t))$ of (2.4) which satisfies $(x(0),z(0),w(0))=(\pi(w),\sigma(w),w)$, for every $\varepsilon>0$ and every $T>0$ also satisfies

$$\|x(t)-\pi(w^0)\|<\varepsilon \quad \|w(t)-w^0\|<\varepsilon$$

at some $t>T$. Since the positive limit set of s is dense in W, then (R) can hold only if (2.5c) holds. This, in turn, yields (2.5b).

For the sufficiency, observe that if (2.5a) and (2.5b) hold, then the graph of $x=\pi(w)$, $z=\sigma(w)$ is a center manifold for (2.4). Since the equilibrium $(x,z,w)=(0,0,0)$ of (2.4) is stable the center manifold is locally attractive and satisfies

$$\|x(t)-\pi(w(t))\|\le Me^{-at}\|x(0)-\pi(w(0))\| \quad (M>0,\ a>0)$$

for all $w(0)$ sufficiently close to zero and all $t\ge0$. Thus, by continuity of $h(x,w)$, $e(t)\to0$ as $t\to\infty$, i.e. (R) is satisfied.

Theorem 1. Under the hypothesis (H1), (H2) and (H3), the Error Feedback Regulator Problem is solvable if and only if there exists $C^k(k\ge2)$ mappings $x=\pi(w)$, with $\pi(0)=0$, and $u=c(w)$, with $c(0)=0$, both defined in a neighborhood $W^o\subset W$ of 0, satisfying the conditions

$$\frac{\partial\pi}{\partial w} s(w) = f(\pi(w),w,c(w)) \tag{2.6a}$$

$$h(\pi(w),w) = 0 \tag{2.6b}$$

Proof of Theorem 1. By (H2), there exists a mapping $k(x)$ which renders the equilibrium $x=0$ of

$$\dot{x} = f(x,0,k(x))$$

exponentially stable. By (H3), there exist mappings $\gamma_1(z_1,z_2,e)$, $\gamma_2(z_1,z_2,e)$ satisfying

$$\gamma_1(z_1,z_2,h(z_1,z_2)) = f(z_1,z_2,0)$$

$$\gamma_2(z_1,z_2,h(z_1,z_2)) = s(z_2)$$

with the property that the equilibrium $(z_1,z_2)=(0,0)$ of

$$\dot{z}_1 = \gamma_1(z_1,z_2,0)$$

$$\dot{z}_2 = \gamma_2(z_1,z_2,0)$$

is exponentially stable.

Suppose $\pi(w)$ and $c(w)$ satisfy (2.6), and consider the error feedback controller

$$u = c(z_2)+k(z_1-\pi(z_2)) = \theta(z_1,z_2)$$

$$\dot{z}_1 = \gamma_1(z_1,z_2,e) + f(z_1,z_2,\theta(z_1,z_2)) - f(z_1,z_2,0)$$

$$\dot{z}_2 = \gamma_2(z_1,z_2,e)$$

This controller satisfies (S), as it can be proven by means of center manifold theory. Moreover, by construction, this controller also satisfies (2.5), with

$$\sigma(w) = \begin{bmatrix} \pi(w) \\ w \end{bmatrix}$$

Thus, by Lemma 1, this controller is such that (R) is also satisfied. The proof of the necessity is a straightforward consequence of Lemma 1.

We stress the fact that the proof of this theorem was constructive.

3. "ROBUST" LINEAR REGULATORS DO NOT TOLERATE NONLINEAR PERTURBATIONS

It is well known that the possibility of achieving the requirement (S), namely the exponential stability of the equilibrium (0,0) of (2.4) (with w set equal to 0), depends entirely on the properties of the linear approximation of (2.1)-(2.2) at $(x,w)=(0,0)$. Since the proof of Lemma 1 shows that, whenever the regulation requirement (S) is achieved, the error converges exponentially to 0, one might conjecture that also the possibility of achieving output regulation might depend only on the linear approximation of (2.1)-(2.2) at $(x,w)=(0,0)$, i.e. that a design based exclusively on the knowledge of the parameters of such a linear approximation might be sufficient to the purpose regulating the output. This conjecture appears to be quite natural, in view of the fact that linear regulators, if properly designed, are known to be "robust", i.e. are such as to impose output regulation also in the presence of unknown perturbations the parameters of the plant as long as the closed loop remains stable (see [5] [7]).

Somehow surprisingly, this conjecture is not true, in the sense that there are cases in which the solution of a nonlinear regulation problem is achievable only by means of a specific nonlinear design. This fact, that will be illustrated in the following elementary example, not only justifies the interest in a regulator theory specifically suited to nonlinear plants, but also underlines a major limitation of the theory of "robust" output regulation for linear systems, namely the fact that the so-called robust regulators tolerate only perturbations which leave unaltered the linear structure. To the best of our knowledge such a limitation has not been addressed before in the appropriate technical literature.

Example. Consider the system

$$\dot{x} = -x+p(x)+u$$
$$y = x$$

where $x,u,y\in R$ and $p(x)$ is some unspecified perturbation. Suppose the regulation task is to track any sinusoidal reference output of the form

$$y_R(t) = M\sin(at+\phi)$$

of fixed frequency a. The problem can be formally stated as in Section 2, by defining an exosystem

$$\begin{bmatrix}\dot{w}_1\\ \dot{w}_2\end{bmatrix} = \begin{bmatrix}aw_2\\ -aw_1\end{bmatrix} = S\begin{bmatrix}w_1\\ w_2\end{bmatrix}$$

and setting

$$e = x-w_1$$

Suppose $p(x)=0$. In this case, elementary theory shows that the following regulator

$$\dot{z}_1 = az_2$$
$$\dot{z}_2 = -az_1+e$$
$$u = Kz_1$$

solves the problem, provided that $a>K>0$. In fact, this choice yields a closed loop system

$$\dot{x} = -x+Kz_1$$
$$\dot{z}_1 = az_2$$
$$\dot{z}_2 = -az_1+x-w_1$$
$$e = x-w_1$$

which is exponentially stable and such that the transfer function

$$\frac{E(s)}{W_1(s)} = -\frac{s^3+s^2+a^2s+a^2}{s^3+s^2+a^2s+a^2-Ka} \qquad (3.1)$$

between the reference w_1 and the error e vanishes at $s=\pm ja$. Thus, output regulation is achieved.

Since the zeros of (3.1) at $\pm ja$ are induced by the corresponding eigenvalues in the compensator, and the latter are not modified by plant parameter variations, a regulator of this type is usually credited to be "robust". However, as we will show in a moment, this is true only if linear perturbations are allowed.

For, suppose $p(x) \neq 0$ and consider the corresponding closed loop system

$$\dot{x} = -x+p(x)+Kz_1$$
$$\dot{z}_1 = az_2$$
$$\dot{z}_2 = -az_1+x-w_1$$
$$e = x-w_1$$

Assuming that $p(x)$ is such that this system is still exponentially stable, we can use Lemma 1 to check whether or not regulation is being achieved. There must exist $x=\pi(w)$, $z_1=\sigma_1(w)$, $z_2=\sigma_2(w)$ such that

$$\frac{\partial \pi}{\partial w} Sw = -\pi(w)+p(\pi(w))+K\sigma_1(w)$$

$$\frac{\partial \sigma_1}{\partial w} Sw = a\sigma_2(w) \tag{3.2}$$

$$\frac{\partial \sigma_2}{\partial w} Sw = -a\sigma_1(w) \tag{3.3}$$

$$0 = \pi(w)-w_1$$

The latter clearly yields

$$\pi(w) = w_1$$

and this, replaced into the first one, yields

$$\sigma_1(w) = \frac{1}{K}\left[aw_2+w_1-p(w_1)\right]$$

So, regulation is achieved if and only if (3.2) and (3.3) can be satisfied.

Now, if $p(x)$ were linear, i.e. $p(x)=px$ then these equations would be satisfied for all p (and this is a confirmation of the "robustness" properties versus any perturbations that keeps unaltered the linear structure). In fact, (3.2) yields in this case

$$\sigma_2(w) = \frac{1}{K}\left[-aw_1+w_2-pw_2\right]$$

which renders (3.3) identically satisfied for all p.

However, if for instance $p(x)$ were quadratic, i.e.

$$p(x) = px^2$$

then, a solution of (3.2) would exist only for $p=0$. In fact, (3.2) yields in this case

$$\sigma_2(w) = \frac{1}{K}\left[-aw_1+w_2-2pw_1w_2\right]$$

and (3.3) becomes

$$-a^2w_2-aw_1-2paw_2^2+2paw_1^2 = -a^2w_2-aw_1-paw_1^2$$

In view of Lemma 1, we conclude that the regulator in question is not robust versus nonlinear perturbations.

On the other hand, it is important to stress that, if the perturbation term were exactly known, the regulator problem would always be solvable (independently on whether or not $p(x)$ is linear in x). Straightforward application of Theorem 1 shows in fact that the corresponding necessary and sufficient conditions are indeed satisfied, since

$$\frac{\partial\pi}{\partial w}Sw = -\pi(w)+p(\pi(w))+c(w)$$
$$0 = \pi(w)-w_1$$

are solved by

$$\pi(w) = w_1$$
$$c(w) = aw_2+w_1-p(w_1)$$

Note that $c(w)$ is nonlinear if so is $p(x)$.

REFERENCES

(1) V. Anantharam, C.A. Desoer, Tracking and Disturbance Rejection of MIMO Nonlinear Systems with a PI or PS Controller, IEEE Conf. Dec. Contr., 24 (1985), pp. 1367-68.

(2) C.I. Byrnes and A. Isidori, Output Regulation of General Nonlinear Systems, C.R. Acad. Sci. Paris (June 1989).

(3) J. Carr, *Applications of Centre Manifold Theory*, Springer Verlag (1981).

(4) E.J. Davison, The Output Control of Linear Time-Invariant Multi-Variable System with Unmeasurable Arbitrary Disturbances, *IEEE Trans. Aut. Contr.*, AC-17 (1972), pp. 621-630.

(5) E.J. Davison, The Robust Control of a Servomechanism Problem for a Linear Time-Invariant Multivariable Systems, *IEEE Trans. Aut. Contr.*, AC-21 (1976), pp. 25-34.

(6) B.A. Francis, The Linear Multivariable Regulator Problems, *SIAM J. Contr. Optimiz.*, 15 (1977), pp. 486-505.

(7) B.A. Francis and W.M. Wonham, The Internal Model Principle for Linear Multivariable Regulators, *J. Appl. Math. Optimiz.*, 2(1975), pp. 170-194.

(8) J. Huang and W.J. Rugh, On a Nonlinear Multivariable Servomechanism Problem, *Technical Report JHU/ECE* - 88/04.2 (1988).

(9) A. Isidori and C.I. Byrnes, Output Regulation of Nonlinear Systems, *IEEE TRans. Aut. Contr.*, AC-35 (February 1990).

A METHOD FOR HIGHER ORDER, DYNAMICAL FEEDBACK IN NONLINEAR SERVOMECHANISM PROBLEMS*

PATRICK M. SAIN, MICHAEL K. SAIN, AND ANTHONY N. MICHEL
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

The use of higher-order spectral information is of considerable current interest in such areas as identification of nonminimum phase systems, sonar and radar signature analysis, texture discrimination, image coding, tomography, analysis of electroencephalograms, array processing and speech enhancement. This paper introduces the idea of using such data in the design of feedback dynamics for nonlinear servomechanisms. The higher-order concepts are embodied in transforms of the kernels of certain Volterra representations. An application to the control of a rotating DC-to-AC converter is presented, in which the design goal is to achieve a partial feedback linearization with associated decoupling. These results generalize those of Al-Baiyat and Sain, 1989, for the case of regulators.

I. INTRODUCTION

Most practical control problems involve plants that are nonlinear to a certain degree. The typical approach to designing control systems for such plants begins with a local linearization in a neighborhood of a nominal operating point. A linear model, valid in that neighborhood, is obtained, and then a convenient linear feedback design is applied. Control in a general region containing several operating points is accomplished by obtaining families of local linear descriptions at desired steady-state conditions and scheduling them as a function of plant operating conditions. In general, however, nonlinear processes can only be adequately characterized by nonlinear models.

In recent years, considerable progress has been made in representing the behavior of nonlinear systems using Volterra series. In the case of multiple-input, multiple-output, finite-dimensional, time-invariant systems such a representation has the form

$$y(t) = \int_0^t p_1(\tau_1)u(t-\tau_1)d\tau_1 + \int_0^t \int_0^{\tau_1} p_2(\tau_1,\tau_2)u(t-\tau_1)\otimes u(t-\tau_2)d\tau_2 d\tau_1 + \dots,\ t \geq \tau_1 \geq \tau_2 \geq \dots \geq 0 \quad (1.1)$$

where p_i, $i=1,2,\dots$, is called the i^{th} Volterra kernel. The goal of this paper is to apply the Volterra series to the higher-order design of nonlinear servomechanisms using a method of feedback synthesis, which addresses the ability to design simultaneously for a specified output response and a reasonable control signal.

Such a feedback synthesis approach, utilizing linear multivariable output feedback, was introduced by Peczkowski, Sain and Leake [1], for nonlinear feedback systems operating at a single point. We shall refer to this method using the acronym PSL. The design is performed using transfer functions obtained from a first-order model of the system at the point of operation. In [2,3,4,5,6,7], PSL was extended to nonlinear servomechanism operation using linear approximations of the plant and operating point schedules. In 1985, Al-Baiyat and Sain [8,9,10] extended the PSL idea itself to involve design at an operating point using higher-order, nonlinear representations of a plant. Here, a finite number of terms in the Volterra series were used to describe the input-output behavior of the nonlinear system operating at a point. Multidimensional Laplace operators were then used to obtain transfer functions of the Volterra kernels in the frequency domain, where extensions of PSL could be applied. In a manner similar to [2], this paper extends the method of Al-Baiyat and Sain to nonlinear servomechanism operation by obtaining families of local, higher-order, nonlinear descriptions at desired steady-state operating points and scheduling them as a function of plant conditions.

---

*This work was supported in part by the Frank M. Freimann Chair in Electrical Engineering at the University of Notre Dame, the Jesse H. Jones Research Fund, and The Ohio State University's Distinguished Visiting Professorship.

II. PROBLEM DEFINITION

Let R, U, and Y denote spaces of requests, plant inputs, and plant outputs, respectively. Let P:U→Y denote a nonlinear plant in an input-output sense. Define the desired closed-loop response to a command by T:R→Y, and the desired plant input for a command by M:R→U. The operators P, T, and M are assumed to have Volterra series representations at operating points within the region of operation for the servomechanism. An illustration of these operators is given in Figure 2.1.

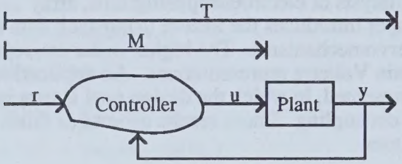


Figure 2.1.

$$\begin{tikzcd}
& R \arrow[dl, dashed, "M"'] \arrow[dr, dashed, "T"] & \\
U \arrow[rr, "P"'] & & Y
\end{tikzcd}$$

Figure 2.2.

The PSL technique divides naturally into two sub-tasks. The first depends only upon plant characteristics, and is concerned with finding a pair (M,T) such that the diagram in Figure 2.2 commutes; that is, $T = P \circ M$. The second is to find a controller which realizes the pair (M,T) in the feedback configuration shown in Figure 2.1. In the discussion which follows, it will be indicated how the desired behavior pair (M,T) can be directly implemented in the controller.

The controller has a twofold role: closing the loop and providing appropriate inputs to the loop. To this end, the controller can be considered to have two parts. The first is a prefilter which splits the request signal into a desired plant output, $y_r \in Y$, and a corresponding desired plant input, $u_r \in U$, and is defined by $C_{pf}:R \to U \times Y$. The second part accomplishes the feedback, and is defined by $C_{fb}:\text{Im}\, C_{pf} \times Y \to U$.

First, consider the prefilter, C_{pf}. As $U \times Y$ admits the direct sum structure $U \oplus Y$, we can construct the diagram in Figure 2.3, where p_u and p_y are the projections of $U \oplus Y$ onto U and Y, respectively, with i_u and i_y being the corresponding injections. From [8, p. 24], we have $C_{pf} = i_u \circ M + i_y \circ T$, with the implication that the nonlinear system C_{pf} with two outputs splits naturally into two nonlinear systems, each having a single output. Without loss of generality, the injective mappings can be incorporated into C_{fb}, resulting in the system shown in Figure 2.4. If we require that $y_r = P(u_r)$, then it follows that $T = P \circ M$.

$$\begin{tikzcd}
& R \arrow[dl, "M"'] \arrow[d, "C_{pf}"] \arrow[dr, "T"] & \\
U \arrow[r, shift left, "i_u"] & U \oplus Y \arrow[l, shift left, "p_u"] \arrow[r, shift right, "p_y"'] & Y \arrow[l, shift right, "i_y"']
\end{tikzcd}$$

Figure 2.3.

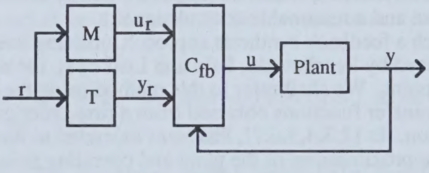


Figure 2.4.

Now consider $u = C_{fb}(u_r,y_r,y)$. If we expand around the operating point y_r using a Taylor series, we obtain

$$\begin{aligned} u &= C_{fb}(u_r,y_r,y_r+(y-y_r)) \\ &= C_{fb}(u_r,y_r,y_r) + DC_{fb}(u_r,y_r,y_r)(y-y_r) + \frac{1}{2!}D^2C_{fb}(u_r,y_r,y_r)(y-y_r)^{(2)} + \dots , \end{aligned} \tag{2.1}$$

where D denotes the differential operator, and $(y-y_r)^{(i)}$ denotes the Cartesian product of $(y-y_r)$

taken i times with itself. Of course, the symmetric bilinear map $D^2C_{fb}(u_r,y_r,y_r)$ can be replaced by a linear map acting on $(y-y_r)\otimes(y-y_r)$, and in a similar manner, the symmetric i-linear map $D^{(i)}C_{fb}(u_r,y_r,y_r)$ can be expressed as a linear map operating on $(y-y_r)\otimes\ldots\otimes(y-y_r)$, where there are i terms in the latter product. If we make the assumption that $u_r = C_{fb}(u_r,y_r,y_r)$, which simply says that the controller produces the desired control when y is the desired output, and define

$$G_r(y_r-y) = -\,DC_{fb}(u_r,y_r,y_r)(y_r-y) + \frac{1}{2!}D^2C_{fb}(u_r,y_r,y_r)(y_r-y)^{(2)} - \ldots, \qquad (2.2)$$

we obtain $u = u_r + G_r(y_r-y)$, resulting in the configuration shown in Figure 2.5. Here, the prefilters M and T work to directly produce the desired responses in u and y, while the feedback loop, with its desired response given by T_L, provides accurate steady-state tracking of the signal y_r. Notice that $G_r(0) = 0$.

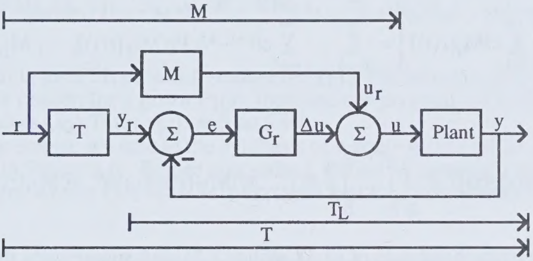


Figure 2.5.

The fundamental idea for synthesis of the nonlinear servomechanism elements is that families of local, higher-order, nonlinear dynamic descriptions taken at desired steady-state operating points may be scheduled as a function of operating conditions to produce nonlinear servo action over a region of interest.

For example, (2.2) shows that the nonlinear operator G_r acting on (y_r-y) is a function of the operating point y_r. Thus, as the request r is changed, the operating point y_r changes in response, resulting in a G_r which changes along with the request. Repeated evaluation of (2.2) at a set of operating points gives rise to a family of controllers $\{G_r\}$. The goal is to link members of the family together to produce a scheduled nonlinear function C_{fb}, with each G_r regarded as giving accurate nonlinear dynamic behavior in some local region of a steady-state point. Under reasonable conditions of smoothness and with enough members in the family, a careful linking typically leads to an accurate and useful C_{fb} over the region of interest.

Similar remarks may be made for M and T.

III. DESIGN WITH VOLTERRA OPERATORS

Recall that the pair (M,T) is to be chosen such that $T = P \circ M$, where the operators T, P, and M admit Volterra representation. Define the homogeneous operator

$$P_i[u(t)] = \int_0^t\int_0^{\tau_1}\cdots\int_0^{\tau_{i-1}} p_i(\tau_1,\ldots,\tau_i)u(t-\tau_1)\otimes u(t-\tau_2)\otimes\ldots\otimes u(t-\tau_i)d\tau_i\ldots d\tau_2 d\tau_1. \qquad (3.1)$$

Using the operator notation, (1.1) can be written as in (3.2a), with (3.2b) and (3.2c) following in a similar manner:

$$y(t) = P[u(t)] = \sum_{i=1}^{\infty} P_i[u(t)], \quad y = T[r(t)] = \sum_{j=1}^{\infty} T_j[r(t)], \quad \text{and } u = M[r(t)] = \sum_{k=1}^{\infty} M_k[r(t)]. \qquad (3.2a,b,c)$$

In order to obtain a relation between the operators p_i, t_j, and m_k, substitute (3.2c) into (3.2a) and equate to (3.2b), obtaining

$$\sum_{j=1}^{\infty} T_j[r(t)] = \sum_{i=1}^{\infty} P_i\left[\sum_{j=1}^{\infty} M_j[r(t)]\right]. \qquad (3.3)$$

If we replace the request signal r(t) by cr(t), where c is an arbitrary constant, and use the fact that Volterra operators are multilinear [11,12,13], we obtain as shown in [8,9],

$$\sum_{j=1}^{\infty} c^j T_j[r(t)] = \sum_{i=1}^{\infty} P_i\left[\sum_{j=1}^{\infty} c^j M_j[r(t)]\right]. \qquad (3.4)$$

Define

$$P_i\left[\sum_{j=1}^{\infty} c^j M_j[r(t)]\right] = \sum_{j_1=1}^{\infty} \cdots \sum_{j_i=1}^{\infty} c^{j_1+\ldots+j_i} P_i(M_{j_1}[r(t)], \ldots, M_{j_i}[r(t)]), \qquad (3.5)$$

where

$$P_i(M_{j_1}[r(t)], \ldots, M_{j_i}[r(t)]) = \int_0^t \int_0^{\tau_1} \cdots \int_0^{\tau_{i-1}} p_i(\tau_1,\ldots,\tau_i) M_{j_1}[r(t-\tau_1)] \otimes \ldots \otimes M_{j_i}[r(t-\tau_i)] d\tau_i \ldots d\tau_1. \qquad (3.6)$$

Then expanding the right-hand side of (3.4), using (3.5), and suppressing the argument [r(t)] yields

$$\sum_{j=1}^{\infty} c^j T_j = \sum_{i=1}^{\infty} \sum_{j_1=1}^{\infty} \cdots \sum_{j_i=1}^{\infty} c^{j_1+\ldots+j_i} P_i(M_{j_1}, \ldots, M_{j_i}). \qquad (3.7)$$

Equating like powers of c in both sides of (3.7) gives

$$T_1 = P_1 M_1, \qquad (3.8a)$$
$$T_2 = P_1 M_2 + P_2(M_1, M_1), \qquad (3.8b)$$
$$T_3 = P_1 M_3 + P_2(M_1, M_2) + P_2(M_2, M_1) + P_3(M_1, M_1, M_1), \qquad (3.8c)$$

with the general term being of the form

$$T_i = P_1 M_i + \sum_{j=2}^{i} \left\{ \sum_{k_1=1}^{i-(j-1)} \; \sum_{k_2=1}^{i-(j-1)-(k_1-1)} \cdots \sum_{k_{j-1}=1}^{i-(j-1)-(k_1-1)-\ldots-(k_{j-2}-1)} P_j(M_{k_1}, M_{k_2}, \ldots, M_{k_{j-1}}, M_{i-k_1-k_2-\ldots-k_{j-1}}) \right\}. \qquad (3.8d)$$

As the equations in (3.8) are convolutional in nature, it is expedient to work in the transform domain. To do this, define the multidimensional Laplace transform of the i^{th} Volterra plant kernel as in [14,15]

$$P_i(s_1,\ldots,s_i) = \int_0^{\infty} \cdots \int_0^{\infty} p_i(\tau_1,\ldots,\tau_i)\, e^{-(s_1\tau_1+\ldots+s_i\tau_i)}\, d\tau_1 \ldots d\tau_i. \qquad (3.9)$$

Applying (3.9) to (3.8),

$$
\begin{aligned}
T_1(s) &= P_1(s)M_1(s), && (3.10a)\\
T_2(s_1,s_2) &= P_1(s_1+s_2)M_2(s_1,s_2) + P_2(s_1,s_2)(M_1(s_1)\otimes M_1(s_2)), && (3.10b)\\
T_3(s_1,s_2,s_3) &= P_1(s_1+s_2+s_3)M_3(s_1,s_2,s_3) + P_2(s_1,s_2+s_3)(M_1(s_1)\otimes M_2(s_2,s_3)) + && (3.10c)\\
&\quad P_2(s_1+s_2,s_3)(M_2(s_1,s_2)\otimes M_1(s_3)) + P_3(s_1,s_2,s_3)(M_1(s_1)\otimes M_1(s_2)\otimes M_1(s_3)),
\end{aligned}
$$

with the general term being given by

$$
T_i(s_1,\ldots,s_i) = P_1(s_1+\ldots+s_i)M_i(s_1,\ldots,s_i) + \qquad (3.10d)
$$

$$
\sum_{j=2}^{i}\left\{\sum_{k_1=1}^{i-(j-1)}\ \sum_{k_2=1}^{i-(j-1)-(k_1-1)} \cdots \sum_{k_{j-1}=1}^{i-(j-1)-(k_1-1)-\ldots-(k_{j-2}-1)} P_j(s_1+\ldots+s_{k_1}, s_{k_1+1}+\ldots+s_{k_1+k_2},\ldots,s_{k_1+k_2+\ldots+k_{j-1}+1}+\ldots+s_i)\right.
$$

$$
\left.(M_{k_1}(s_1,\ldots,s_{k_1})\otimes M_{k_2}(s_{k_1+1},\ldots,s_{k_1+k_2})\otimes\ldots\otimes M_{i-k_1-\ldots-k_{j-1}}(s_{k_1+k_2+\ldots+k_{j-1}+1},\ldots,s_i))\right\}.
$$

Observe that (3.10a) is the PSL equation studied in [1] by Peczkowski, Sain, and Leake. If the pair $(M_1(s),T_1(s))$ is chosen for a given $P_1(s)$, then one can proceed to (3.10b), where the only unknown pair is $(M_2(s_1,s_2),T_2(s_1,s_2))$, and so forth.

To complete the design, we turn to the synthesis of a controller G which synthesizes a pair (M_L,T_L) as shown in Figure 2.6. Before proceeding, define an operator $E{:}R\rightarrow Y$, Y being the space of output errors, and let $G{:}Y\rightarrow U$. It is assumed that E admits Volterra representation.

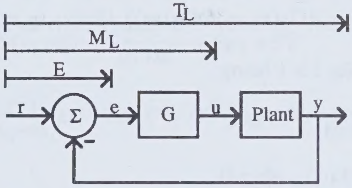


Figure 2.6.

From Figure 2.6, we see that $M_L = G \circ E$, which is similar in form to Figure 2.2, $T = P \circ M$. By inspection, therefore, we can write, after suppressing the subscript L for simplicity,

$$
\begin{aligned}
M_1(s) &= G_1(s)E_1(s), && (3.11a)\\
M_2(s_1,s_2) &= G_1(s_1+s_2)E_2(s_1,s_2) + G_2(s_1,s_2)(E_1(s_1)\otimes E_1(s_2)), && (3.11b)\\
M_3(s_1,s_2,s_3) &= G_1(s_1+s_2+s_3)E_3(s_1,s_2,s_3) + G_2(s_1,s_2+s_3)(E_1(s_1)\otimes E_2(s_2,s_3)) + && (3.11c)\\
&\quad G_2(s_1+s_2,s_3)(E_2(s_1,s_2)\otimes E_1(s_3)) + G_3(s_1,s_2,s_3)(E_1(s_1)\otimes E_1(s_2)\otimes E_1(s_3)),
\end{aligned}
$$

with the general term given as

$$
M_i(s_1,\ldots,s_i) = G_1(s_1+\ldots+s_i)E_i(s_1,\ldots,s_i) +
$$

$$
\sum_{j=2}^{i}\left\{\sum_{k_1=1}^{i-(j-1)}\ \sum_{k_2=1}^{i-(j-1)-(k_1-1)}\ \sum_{k_{j-1}=1}^{i-(j-1)-(k_1-1)-\ldots-(k_{j-2}-1)} G_j(s_1+\ldots+s_{k_1}, s_{k_1+1}+\ldots+s_{k_1+k_2},\ldots,s_{k_1+k_2+\ldots+k_{j-1}+1}+\ldots+s_i)\right.
$$

$$
\left.(E_{k_1}(s_1,\ldots,s_{k_1})\otimes E_{k_2}(s_{k_1+1},\ldots,s_{k_1+k_2})\otimes\ldots\otimes E_{i-k_1-\ldots-k_{j-1}}(s_{k_1+k_2+\ldots+k_{j-1}+1},\ldots,s_i))\right\}. \qquad (3.11d)
$$

Again, from Figure 2.6, we note $e = r - y$, which can be written

$$\sum_{i=1}^{\infty} E_i[r(t)] = I[r(t)] - \sum_{i=1}^{\infty} T_i[r(t)] \tag{3.12}$$

where I is the identity operator. Following a similar procedure as given above, we obtain

$$E_1(s) = I - T_1(s); \;\; E_i(s_1,\ldots,s_i) = -T_i(s_1,\ldots s_i), \;\; i > 1. \tag{3.13a,b}$$

Clearly, from (3.10-3.11) and (3.13), we can find a direct relation between the Volterra kernels of the operators T, M, and G, if the design is well posed on the first-order, or linear, level. This means that $E_1(s)$ has an inverse.

IV. PARTIAL FEEDBACK LINEARIZATION

An interesting special case of the feedback design we proposed in the preceding section is the following. Suppose we want to design a feedback system for a given nonlinear system such that $T_i(s_1,\ldots,s_i) = 0$ for a finite number of i greater than one. We call this a partial linearization.

Suppose that T_2 and T_3 are zero. Then from (3.13) this means that $E_2 = 0$ and $E_3 = 0$. From this, and assuming $E_1(s)$ and $P_1(s)$ to be invertible, it can be shown using (3.10), (3.11), and (3.13) that the first three kernels of the controller G are given by

$$G_1(s) = P_1^{-1}(s)T_1(s)(I-T_1(s))^{-1}, \tag{4.1a}$$

$$G_2(s_1,s_2) = -P_1^{-1}(s_1+s_2)P_2(s_1,s_2)(G_1(s_1)\otimes G_1(s_2)), \tag{4.1b}$$

$$\begin{aligned} G_3(s_1,s_2,s_3) = -P_1^{-1}(s_1+s_2+s_3)\{&P_2(s_1,s_2+s_3)(G_1(s_1)\otimes G_2(s_2,s_3)) + \\ &P_2(s_1+s_2,s_3)(G_2(s_1,s_2)\otimes G_1(s_3)) + P_3(s_1,s_2,s_3)(G_1(s_1)\otimes G_1(s_2)\otimes G_1(s_3))\}, \end{aligned} \tag{4.1c}$$

with the general expression for i > 1 being

$$\begin{aligned} G_i(s_1,\ldots,s_i) &= -P_1^{-1}(s_1+\ldots+s_i) \\ &\sum_{j=2}^{i} \Bigg\{ \sum_{k_1=1}^{i-(j-1)} \; \sum_{k_2=1}^{i-(j-1)-(k_1-1)} \cdots \sum_{k_{j-1}=1}^{i-(j-1)-(k_1-1)-\ldots-(k_{j-2}-1)} P_j(s_1+\ldots+s_{k_1}, s_{k_1+1}+\ldots+s_{k_1+k_2},\ldots,s_{k_1+k_2+\ldots+k_{j-1}+1}+\ldots+s_i) \\ &(G_{k_1}(s_1,\ldots,s_{k_1})\otimes G_{k_2}(s_{k_1+1},\ldots,s_{k_1+k_2})\otimes\ldots\otimes G_{i-k_1-\ldots-k_{j-1}}(s_{k_1+k_2+\ldots+k_{j-1}+1},\ldots,s_i)) \Bigg\}. \end{aligned} \tag{4.1d}$$

V. EXAMPLE

To illustrate the foregoing ideas, we will design a nonlinear servomechanism for the simplified model of a DC-to-AC converter shown in Figure 5.1.

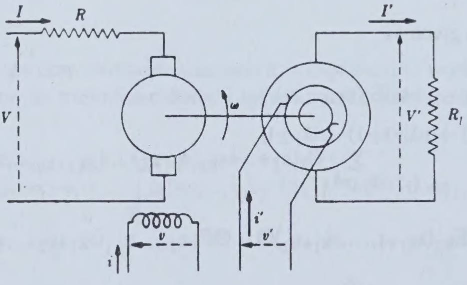


Figure 5.1.

Specifically, we will study the influence of the inputs v' and V on the outputs ω and V' as the system follows a requested operating line, while v is held constant. A description of the model is given by the differential equations [16]

$$\frac{d\omega}{dt} = \frac{1}{J}\left\{k\,\frac{v}{Rr}\left[V - k\,\frac{v}{r}\,\omega\right] - \frac{(V')^2}{\omega R_l}\right\} \tag{5.1a}$$

$$\frac{dV'}{dt} = \frac{V'}{\omega}\frac{d\omega}{dt} - \frac{r'V'}{l'} + \frac{k'}{l'}\,\omega v' \tag{5.1b}$$

where V and V' are the armature and load voltages, v and v' are the respective field voltages of the armature and generator, ω is the angular shaft velocity, J is a moment of inertia, k and k' are emf/torque constants, R and R_l are the armature and load resistances, r and r' are the respective field resistances of the motor and generator, and l' is the field inductance of the generator. We shall assume that J = 1, k = k' = 1, r = r' = 1, R = .2, R_l = 1, l' = .5, and v = 1. If we let $\bar{\omega}$, $\bar{V}$, $\bar{V}'$, and $\bar{v}'$ be the steady-state values of ω, V, V', and v' at an operating point, and if we define

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \omega-\bar{\omega} \\ V'-\bar{V}' \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} V-\bar{V} \\ v'-\bar{v}' \end{bmatrix},$$

then (5.1) becomes

$$\dot{x}_1 = -5(x_1+\bar{\omega}) - \frac{(x_2+\bar{V}')^2}{x_1+\bar{\omega}} + 5u_1 + 5\bar{V}, \tag{5.2a}$$

$$\dot{x}_2 = \frac{x_2+\bar{V}'}{x_1+\bar{\omega}}\,\dot{x}_1 + 2(x_1\bar{v}' + \bar{\omega}\bar{v}' - x_2 - \bar{V}' + \bar{\omega}u_2 + x_1u_2). \tag{5.2b}$$

This system is of the form

$$\dot{x} = f(x) + g(x)u, \tag{5.3a}$$

$$y = h(x), \tag{5.3b}$$

where f, g, and h are analytic functions in a suitable region. As the input enters linearly, such systems are termed linear analytic.

Suppose it is desired to design a nonlinear servomechanism for the DC-to-AC converter with the loop response

$$T_{L,1}(s) = \begin{bmatrix} \dfrac{1}{(\tau_1 s+1)(\tau_2 s+1)} & 0 \\ 0 & \dfrac{1}{(\tau_3 s+1)(\tau_4 s+1)} \end{bmatrix}, \qquad T_{L,2}(s_1,s_2) = [0]_{2x4}, \tag{5.4a,b}$$

with $T_1(s)$ having the same form as $T_{L,1}(s)$.

The first step in the design method is to calculate the kernels of the plant in the transform domain. For this example, the works by Brockett [17], Rugh [13], and Al-Baiyat and Sain [8,9,10] are used. In [17], Brockett shows that a bilinear approximation to a single-input, single-output linear analytic system can be constructed using the linearization idea of Carleman as described by Krener in [18]. Al-Baiyat generalized Brockett's work to the multi-input, multi-output case in [8].

The bilinearization process utilizes a power series expansion of the linear analytic system about an operating point, in a manner similar to (2.1),

$$\dot{x} = \sum_{i=1}^{\infty} A_{1i}x^{i} + \sum_{i=1}^{\infty} D_{1i}(x^{i} \otimes u) + D_{10}u, \quad x(0) = x_0, \qquad (5.5a)$$

$$y = \sum_{i=1}^{\infty} C_{1i}x^{i}, \qquad (5.5b)$$

where the coefficient matrices A_{1i}, C_{1i}, D_{1i}, and D_{10}, $i = 1, 2, \ldots$, are functions of the operating point and x^i is the tensor product of x with itself, i times. As shown in [8], such an expansion can be approximated by a bilinear system of the form

$$\dot{\tilde{x}} = A\tilde{x} + D(\tilde{x} \otimes u) + Bu, \quad x(0) = x_0, \qquad (5.6a)$$

$$y = C\tilde{x}, \qquad (5.6b)$$

where $\tilde{x}$ is a column vector having i entries, the first being x, the second x^2, and so on up to x^i.

Rugh gives an algorithm in [13] for obtaining the Volterra series for single-input, single-output bilinear systems. This is generalized by Al-Baiyat in [8] to include the class of multi-input, multi-output bilinear systems. In addition, Al-Baiyat gives expressions for the Laplace transforms of the Volterra kernels, given below as

$$P_1(s) = C(sI-A)^{-1}B, \qquad (5.7a)$$

$$P_2(s_1,s_2) = C[(s_1+s_2)I-A]^{-1}D\{[(s_1I-A)^{-1}B]\otimes I_m\}, \qquad (5.7b)$$

$$P_3(s_1,s_2,s_3) = C[(s_1+s_2+s_3)I-A]^{-1}D\{[((s_1+s_2)I-A)^{-1}D\{[(s_1I-A)^{-1}B]\otimes I_m\}]\otimes I_m\}, \qquad (5.7c)$$

and, in general,

$$P_i(s_1,s_2,\ldots,s_i) = C[(s_1+s_2+\ldots+s_i)I-A]^{-1}D\{\ldots[((s_1+s_2)I-A)^{-1}D\{[(s_1I-A)^{-1}B]\otimes I_m\}]\ldots\otimes I_m\}, \qquad (5.7d)$$

where m is the number of inputs for the system.

Once the plant kernels are computed in the transform domain, the higher-order controllers can be found using (4.1). Due to the space limitations, the calculations are not shown here. It is important to note, however, that the kernels of G are functions of the operating point $(\overline{\omega}, \overline{V}')$, and that as the request drives the servomechanism along an operating line, the kernels of G will be scheduled correspondingly.

In order to check the design, computer simulations for the closed-loop system were performed. Figures 5.2 and 5.3 show responses for a request submitted to the servomechanism which drives the plant along an operating line where the dynamic behavior changes rapidly.

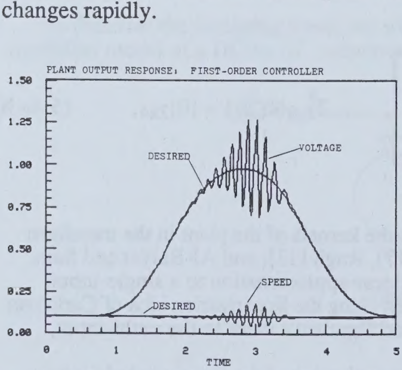


Figure 5.2.

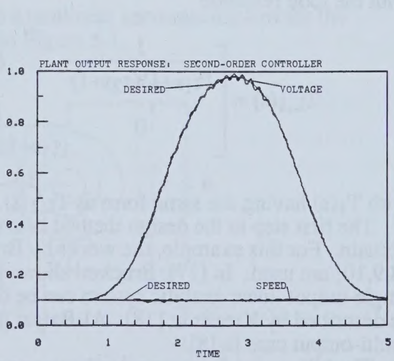


Figure 5.3.

In Figure 5.2, a first-order controller was used, while a second-order controller was used in Figure 5.3. The request was deliberately chosen as a means of illustrating the ability of a higher-order controller to produce desirable results. In both simulations, the desired time constants for $T_1(s)$ were $\tau_1 = \tau_3 = .1$ and $\tau_2 = \tau_4 = .2$, with the loop time constants in $T_{L,1}(s)$ being ten times smaller.

In Figures 5.4 and 5.5, a comparison is made of the error signal, e, for the same request as before. Here, however, the desired loop time constants in $T_{L,1}(s)$ have been changed to $\tau_1 = \tau_3 = .002$ and $\tau_2 = \tau_4 = .005$ in order to respond faster to the dynamic behavior of the plant along the requested operating line. The time constants in $T_1(s)$ remained the same as above. Note that the error signal in Figure 5.5, where a second-order controller was used, is about half the magnitude of the error signal for the first-order controller, shown in Figure 5.4.

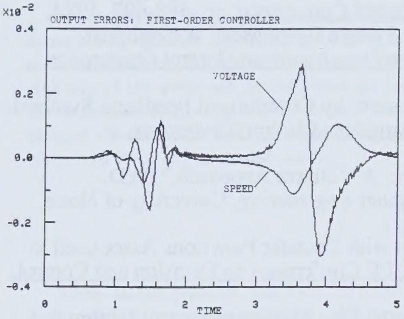


Figure 5.4.

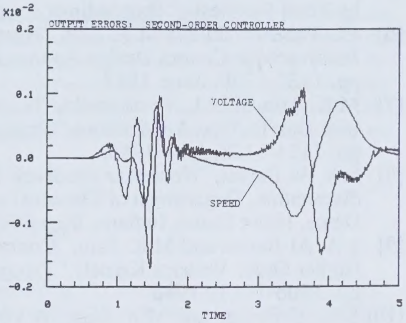


Figure 5.5.

The first output of the second-order controller is shown in Figure 5.6. Note that the magnitude of the output from the second-order kernel of the controller is significant with respect to that of the first-order kernel. The relative magnitude of the total controller output to the plant input and to the output of the prefilter M is shown in Figure 5.7, where we see that controller action is of a reasonable magnitude relative to the size of the plant input.

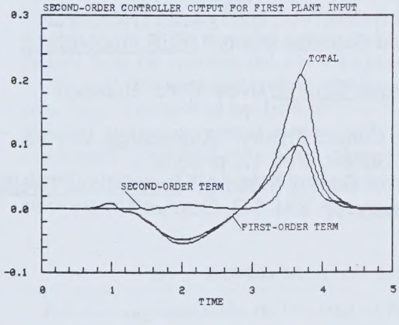


Figure 5.6.

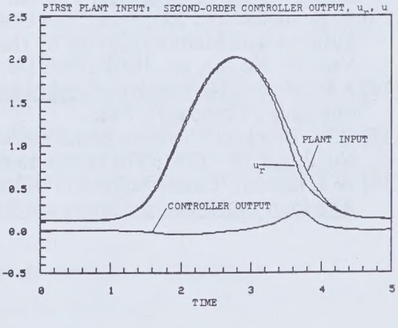


Figure 5.7.

VI. CONCLUSION

This paper has presented a procedure for designing a nonlinear servomechanism using higher-order spectral information about a nonlinear plant. This is an extension of the work done by Al-Baiyat and Sain [9,10], which utilized Volterra series in the PSL method to design higher-order nonlinear regulators. The new procedure is relatively simple to understand, and is a systematic approach. The example has illustrated the potential advantage of using higher-order controllers for nonlinear systems.

VII. REFERENCES

[1] J.L. Peczkowski, M.K. Sain, and R.J. Leake, "Multivariable Synthesis with Inverses," Proceedings Eighteenth Joint Automatic Control Conference, pp. 275-380, June 1979.

[2] J.L. Peczkowski and M.K. Sain, "Scheduled Nonlinear Control Design for a Turbojet Engine," Proceedings IEEE International Symposium on Circuits and Systems, pp. 248-251, April 1981.

[3] M.K. Sain and J.L. Peczkowski, "An Approach to Robust Nonlinear Control Design," Proceedings Twentieth Joint Automatic Control Conference, No. FA-3D, June 1981.

[4] M.K. Sain and J.L. Peczkowski, "Nonlinear Multivariable Design by Total Synthesis," Proceedings American Control Conference, pp. 252-260, June 1982.

[5] J.L. Peczkowski and M.K. Sain, "Design of Nonlinear Multivariable Feedback Controls by Total Synthesis," Proceedings American Control Conference, pp. 688-697, 1984.

[6] J.L. Peczkowski and M.K. Sain, "Synthesis of System Responses: A Nonlinear Multivariable Control Design Approach," Proceedings American Control Conference, pp. 1322-1329, June 1985.

[7] M.K. Sain and J.L. Peczkowski, "Nonlinear Control by Coordinated Feedback Synthesis with Gas Turbine Applications," Proceedings American Control Conference, pp. 1121-1128, June 1985.

[8] S.A. Al-Baiyat, "Nonlinear Feedback Synthesis: A Volterra Approach," Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, Indiana, May 1986.

[9] S.A. Al-Baiyat and M.K. Sain, "Control Design with Transfer Functions Associated to Higher Order Volterra Kernels," Proceedings IEEE Conference on Decision and Control, pp. 1306-1311, 1986.

[10] S.A. Al-Baiyat and M.K. Sain, "A Volterra Method for Nonlinear Control Design," Preprints IFAC Symposium on Nonlinear Control System Design, pp. 76-81, June 1989.

[11] W.A. Porter, "An Overview of Polynomic System Theory," IEEE Proceedings, Vol. 64, No. 1, pp. 18-23, January 1976.

[12] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, (New York: John Wiley), 1980.

[13] W.J. Rugh, Nonlinear system Theory: The Volterra/Wiener Approach, (Baltimore: The Johns Hopkins University Press), 1981.

[14] J.J. Bussgang, L. Ehrman, and J. Graham, "Analysis of Nonlinear Systems with Multiple Inputs," IEEE Proceedings, Vol. 62, No. 8, pp. 1088-1119, August 1974.

[15] E. Bedrosian and S.O. Rice, "The Output Properties of Volterra Systems (Nonlinear Systems with Memory) Driven by Harmonic and Gaussian Inputs," IEEE Proceedings, Vol. 59, No. 12, pp. 1688-1707, December 1971.

[16] J. Peschon, Disciplines and Techniques of Systems Control, (New York: Blaisdell Publishing Company), 1965.

[17] R.W. Brockett, "Volterra Series and Geometric Control Theory," Automatica, Vol. 12, No. 2, pp. 167-176, 1976 (addendum with E. Gilbert, Vol. 12, p. 635).

[18] A. J. Krener, "Linearization and Bilinearization of Control Systems," Proceedings Twelfth Allerton Conference on Circuit and System Theory, pp. 834-843, October 1974.

THE POSTURAL STABILITY IN BIPED DURING SITTING DOWN MOTION

HOO DENNIS ONG
HOOSHANG HEMAMI
Department of Electrical Engineering, Ohio State University, Columbus, Ohio 43210.

SHELDON SIMON
Department of Orthopedic Surgery, Ohio State University Hospitals, Columbus, Ohio 43210.

ABSTRACT

In this paper, it is proposed that, in well-learned motion, local position and velocity feedbacks are enough to guarantee postural stability. It is shown that the needed position and velocity feedbacks are available from the dynamics of the muscles. Lyapunov's second method is employed to prove the stability of the proposed control strategy for both the constrained and the unconstrained motions. To verify the effectiveness of the control strategy, computer simulations are performed, in which the natural viscosity and stiffness characteristics are modulated by CNS for sitting on a chair, and are compared with experimental results.

I. INTRODUCTION

Some recent research has been devoted to understanding the control strategy humans utilize during different movements [3,5,6,7,12] However, much remains to be known about the functions of the brain, the spinal cord and the proprioceptive and visual systems in controlling movements. In spite of this gap of knowledge, both qualitative and quantitative models of human control strategy are highly desirable. In this paper, a potential control strategy for postural stability is proposed. The potential applications of this reseach are in functional electrical stimulation(FES) and in the control of artifical limbs [1].

A local velocity and position feedback strategy is proposed here that will encompass both the feedbacks of the muscles and the central nervous system(CNS). It is shown that the natural apparent viscosity of the muscular system and some deliberate gravity compensation are sufficient for stability. Further, it is shown that for specific tasks, these natural feedback paths are influenced by the CNS for trajectory planning. Therefore, the control strategy proposed here uses existing apparent viscoelasticity of the muscular system and augments it by additional position and velocity feedback derived from the spindles and whose gains are modulated by the CNS.

To verify the proposed control strategy, it is applied to a well-learned motion, i.e. sitting down on a chair. This motion has both the unconstrained phase of lowering the body and the constrained phase of making permanent contact with the chair. The stability of the system both in the unconstrained phase and constrained phase is shown by the Lyapunov's second method. To verify the effectiveness of the control strategy, computer simulations are performed to show stability and tracking ability.

II. BIPED MODEL AND MOTOR CONTROL SYSTEM

This work was supported by the U.S. Dept. of Education Grant No. H133E80017.

The six-link biped model is shown in figure 1. The six links correspond respectively to the leg, the thigh, the torso, the upper arm, the lower arm and the hand. The parameters of weight, height and moments of inertia are taken from [9]. The angles at the joints $\Theta = [\Theta_1, \ldots, \Theta_6]$ and $\dot{\Theta} = \frac{d\Theta}{dt}$ constitute the 12 states of the system. The equations of the unconstrained n-link biped are:

$$I(\Theta)\ddot{\Theta} + B(\Theta, \dot{\Theta}) + G(\Theta) = E(\Theta)U \tag{1}$$

where Θ: vector of angles at the joint, $I(\Theta)$: inertia matrix, $B(\Theta, \dot{\Theta})$: centrifugal and coriolis terms, $G(\Theta)$: the gravity vector, $E(\Theta)$: torque input map, and U: applied joint torques.

For the constrained motion of the biped, the equations of motion are:

$$I(\Theta)\ddot{\Theta} + B(\Theta, \dot{\Theta}) + G(\Theta) = E(\Theta)U + \frac{\partial C^T}{\partial \Theta} F \tag{2}$$

where F is a single point contact force vertical to the chair and C is the constraint equation describing the contact.

The functions of the CNS and the sensory feedbacks in controlling movements have attracted considerable interest. The simplified human motor systems, as shown in figure 2, can quantitatively show the interactions of the nervous system and the muscle. There are four inputs to the bipedal system which are neural inputs, apparent passive viscoelasticity and active programmed viscoelasticity, and the intended learned movement. The neural input compensates for the gravity. The apparent passive viscoelasticity provides the necessary feedback for stability. The active programmed viscoelasticity gives the desired speed for a particular motion. The intended learned movement supplies the muscles length and velocities for the intended movement. Each input has its own unique functions and they all combine together to provide the agility of human movements. The velocity feedback results from the apparent viscosity of the muscle [16]. This can be observed from the force-velocity relationship of a muscle. The force decreases as the velocity increases. A simple approximation is that the force decreases linearly as the velocity increases. Therefore, the apparent viscosity can be modelled by negative velocity feedback. The spindle provides position feedback. Therefore, the spindle can be seen as a position feedback of which the gain can be changed by the higher centers through the gamma efferent nerve. The block diagram for the above system with position and velocity feedbacks is shown as in figure 3. The total input torques to the biped system are the neural input, M, position feedback, k_p, and velocity feedback, k_v:

$$E(\Theta)U = M - k_p(\Theta - \Theta_0) - k_v\dot{\Theta} \tag{3}$$

where

$$M = -\frac{\partial L}{\partial \Theta}^T N, k_p = (\frac{\partial L}{\partial \Theta}^T)K(\frac{\partial L}{\partial \Theta}), k_v = (\frac{\partial L}{\partial \Theta}^T)B(\frac{\partial L}{\partial \Theta}) \tag{4}$$

and B and K are apparent viscous matrix and stiffness matrix of all the muscles, L is a vector of the lengths of all the muscles involved, and Θ_0 is the desired equilibrium point.

III. THE STABILITY OF THE SITTING DOWN MOTION BY LYAPUNOV'S SECOND METHOD

The effect of the position and velocity feedback of equation (3) on stability is considered in this section. The unconstrained phase is when the biped starts to sit and before touches the chair. The constrained phase is the motion when the biped touches the chair and onwards. In this study, the stability of the biped using linear(position and velocity) feedback is proven by Lyapunov's second method both in the constrained and the unconstrained phases [8,13,17].

To prove stability, let k_p and k_v be constant and not depend on Θ. Further let k_p be selected such that the equilibrium point Θ_0 is a minimal point for the potential energy of the system. This means gravity is compensated by storing energy in stretched muscles. Let a Lyapunov function $V(\Theta, \dot\Theta)$ be defined as an energy function consisting of the sum of the kinetic energy, potential energy and the energy stored in the stretched muscles:

$$Q = \frac{1}{2}\dot\Theta^T I(\Theta)\dot\Theta + v(\Theta) - v(\Theta_0) - G^T|_{\Theta_0}(\Theta - \Theta_0) + \frac{1}{2}(\Theta - \Theta_0)^T k_p(\Theta - \Theta_0) \tag{5}$$

where $v(\Theta)$ is the potential energy of the biped system. It follows that $Q(\Theta_0, 0) = 0$ and select k_p such that

$$v(\Theta) - v(\Theta_0) - G^T|_{\Theta_0}(\Theta - \Theta_0) + \frac{1}{2}(\Theta - \Theta_0)^T k_p(\Theta - \Theta_0) > 0 \tag{6}$$

It can be shown that if the system is disturbed from equilibrium

$$\dot{Q} = -\dot\Theta^T k_v \dot\Theta < 0 \tag{7}$$

and the system is driven to its unique equilibrium point, Θ_0 where $Q = 0$. Let's define M(Θ) to be equal to equation (6). M(Θ) is larger than zero if M(Θ) is a convex function. To show that, by differentiating M(Θ) with respect to Θ,

$$\frac{\partial M}{\partial \Theta}^T = \frac{\partial v}{\partial \Theta}^T - G|_{\Theta_0} + k_p(\Theta - \Theta_0) \tag{8}$$

and

$$M(\Theta_0) = 0, \frac{\partial M}{\partial \Theta}^T(\Theta_0) = 0 \tag{9}$$

Futhermore, differentiating equation (8) yields and if k_p is chosen such that

$$\frac{\partial^2 M}{\partial \Theta^2} = \frac{\partial^2 v}{\partial \Theta^2} + k_p \tag{10}$$

If k_p is chosen such that

$$\frac{\partial^2 v}{\partial \Theta^2} + k_p > 0 \tag{11}$$

M(Θ) becomes a strictly convex function, and has a global minimum M(Θ_0) = 0 at $\Theta = \Theta_0$, therefore ensuring that M(Θ) is larger than zero for all Θ. Therefore, if equation (11) is satisfied, global stability for unconstrained stage is gauranteed.

The constrained phase consists of two motion: the collision between the biped and the chair, and the motion of the biped when it is sitting on the chair. Assumption of an impactive collision with the chair is unrealistic, and therefore it is assumed that a compressible tissue interfaces between the contact point and the chair [2,4]. Since the compressible tissue are viscoelastic, the contact point here is modelled as a spring and dashpot in parallel. It is assumed here that the frictional force at the point of contact is negligible. This is true for this simple sitting down motion. Therefore, the contact force has only normal direction but no tangential direction. The viscoelastic force, F, that is exerted at the spring and dashpot combination is:

$$F = k_s C + k_d \dot{C} \tag{12}$$

where k_s is the spring constant and k_d is the dashpot constant are both positive. The constraint, C, is a holonomic constraint that requires the biped not to penetrate the chair. The constraint can be represented as:

$$C(\Theta) = d - l_1 cos(\Theta_1) + l_2 cos(\Theta_2) \tag{13}$$

where d is the height of the chair. Assuming that the leg and thigh would not move substantially after the biped makes contact with the chair, the constraint, C, can be expressed in the vicinity of the operating point Θ_0 as:

$$C(\Theta) = C(\Theta_0) + \frac{\partial C}{\partial \Theta}|_{\Theta_0}(\Theta - \Theta_0) \tag{14}$$

$$\frac{d}{dt}(C(\Theta)) = \frac{\partial C}{\partial \Theta}|_{\Theta_0}\dot{\Theta} \tag{15}$$

Therefore, the viscoelastic force, F, can be approximated by:

$$F = -k_s(C(\Theta_0) + \frac{\partial C}{\partial \Theta}|_{\Theta_0}(\Theta - \Theta_0)) - k_d \frac{\partial C}{\partial \Theta}|_{\Theta_0}\dot{\Theta} \tag{16}$$

The values of the spring constants is determined from the sitting-down experiment [9] by the least square method as:

$$k_s = \begin{cases} 3200.2N/m & 0 < C \leq .023656m \\ 17619N/m & .023656m < C \end{cases} \tag{17}$$

and k_d is set to be 0.1 times of the k_s value.

Let's define the input, U, to be:

$$EU = -k_p(\Theta - \Theta_0) - k_v(\dot{\Theta}) + G|_{\Theta_0} + \frac{\partial C}{\partial \Theta}^T k_s C(\Theta_0) \tag{18}$$

There is an additional term in the input when compared with equation (3) of the unconstrained phase. The additional term is to compensate for the contact force. Using the same assumption that the leg and the thigh angles will not change substantially after contact, the following approximation can be made:

$$\frac{\partial C}{\partial \Theta}^T \approx \frac{\partial C}{\partial \Theta}^T |_{\Theta_0} \tag{19}$$

The Lyapounov function is defined here as before with an additional term

$$\hat{Q} = \frac{1}{2}\dot{\Theta}^T I(\Theta)\dot{\Theta} + v(\Theta) - v(\Theta_0) - G^T|_{\Theta_0}(\Theta - \Theta_0) + \frac{1}{2}(\Theta - \Theta_0)^T(k_p + \frac{\partial C}{\partial \Theta}|^T_{\Theta_0} k_s \frac{\partial C}{\partial \Theta}|_{\Theta_0})(\Theta - \Theta_0) \tag{20}$$

That additional term is the potential energy of the contact point. The term k_s is selected such that

$$v(\Theta) - v(\Theta_0) - G^T|_{\Theta_0}(\Theta - \Theta_0) + \frac{1}{2}(\Theta - \Theta_0)^T(k_p + \frac{\partial C}{\partial \Theta}^T |_{\Theta_0} k_s \frac{\partial C}{\partial \Theta}|_{\Theta_0})(\Theta - \Theta_0) > 0 \tag{21}$$

It can be shown that

$$\dot{\hat{Q}} = -\dot{\Theta}^T(k_v + \frac{\partial C}{\partial \Theta}^T k_d \frac{\partial C}{\partial \Theta}|_{\Theta_0})\dot{\Theta} < 0 \tag{22}$$

then the system will be driven to its unique equilibrium point by any disturbance. Let's define equation (21) to be $\hat{M}(\Theta)$. With the same philosophy as with the unconstrained system, $\hat{M} > 0$ for all Θ if

$$\frac{\partial^2 v}{\partial \Theta^2} + k_p + \frac{\partial C}{\partial \Theta}^T |_{\Theta_0} k_s \frac{\partial C}{\partial \Theta}|_{\Theta_0} > 0 \tag{23}$$

This will ensure that the system converges to desired Θ_0. Therefore, the biped will converge to the desired Θ_0 if equations (11) and (23) are satisfied respectively for unconstrained and constrained phases.

We have shown above that the elasticity of the muscular system and the natural tendency of the muscles to generate less force under faster movements are the natural stabilizing position and velocity feedback components. The central nervous system, however, may increase or decrease these components by additional positive or negative spindle feedback for different tasks.

Regarding the latter complementary feedback, two kinds of broad structure are envisioned here. These structures relate to the form of the CNS feedback. The first structure implies that the feedback to a muscle comes from many other muscles. The second structure implies that the major component of the feedback to an agonist-antagonist pair comes from the same pair of muscles, i.e. the feedback matrices are dominant in their diagonal elements.

It is hypothesized by Nashner and McCollum [14] that postural controls are organized to use a minimum number of muscles. The diagonal structure above is compatible with the minimum hypothesis of Nashner and McCollum in two ways:

1. In a n-link systems, at most n pairs of muscles are needed to guarantee positive definiteness of the position and velocity gain matrices.
2. If the natural viscoelasticity of some muscle pairs are adequate for the task at hand, a smaller subset of n pairs of active muscles may be involved.

For some daily routine movements [10,11,18], proprioceptive feedback may not be necessary suggesting that the major feedback for those movements can come from the same link. From the above evidence, diagonal feedback may be more likely for some daily motion and it is proposed here that diagonal feedback be employed. The minimum value of position gain, k_p, that can ensure the stability of the biped both in the unconstrained and the constrained phases are calculated from equations (11) and (23) and are found to be the same value. The elements of the diagonal matrix k_p are (248.55,216.85,120.21,12.529,4.9128,0.51503).

IV. COMPUTER SIMULATIONS AND EXPERIMENTS

In the the following simulations, a single joint torque actuator is assumed to be present at every joint. The joint torques could be either positive or negative and account for a pair of muscles at that joint. The experimental data are taken from Kallel [9]. The experiment consisted of recording the motion of a healthy subject as he sat down on a chair and stood up from the chair. The recorded data were the kinematics of the body segments, the chair reaction force and the ground reaction force.

For the first simulation, the biped is commanded to move from the initial position (Θ_i = (5.73°,-7.44°,8.59°,-170.74°, 167.88°,181.91°)) of standing to the equilibrium position of sitting (Θ_f = (32.08°, -89.09°,31.51°,-182.20°,167.88°, 171.89°)). For the unconstrained and constrained phases, the position feedback gains are set to be the minimum gain required for stability ,and the velocity feedback gains are 0.1 times of the position feedback gains. The trajectory of motion of the end-point of the second link are shown in figure 4. The biped hits the chair at 0.39 sec. The biped bounces first downward and then upward before settling down to the desired value. The overshooting occurs because the biped is not commanded to slow down before it hits the chair. The chair reaction force F, as shown in figure 5, is positive all the time after the contact indicating that the biped remains in contact with the chair after sitting. The biped is stable both in the constrained and unconstrained phases.

In the second simulation, the input to the biped system is the angular positions measured in the experiment. Therefore, the biped is commanded to track the angular position trajectory from the experiment. The position feedback gain is also set to the minimum necessary for stability ,and velocity feedback gain is 0.1 times the position gain in the unconstrained stage. The purpose of this simulation is to verify whether the biped can only use the angular position information in tracking the sitting down motion. Moreover, three position and velocity feedback gains in the constrained phase are employed to verify whether the biped uses different gains in sitting down. The position feedback gains used are respectively one time(gain=1), two times(gain=2), and four times(gain=4) the minimum needed for stability. The velocity gains are 0.1 times the position feedback gains used. As can be seen from the graph of the end-point of the second link in figure 6, the biped sits on the chair with less overshoot but eventually leaves the chair after 0.8 seconds as the gain=4. This also can be seen in the chair reaction force as shown in figure 7. As the gains increases, the overshoot decreases. However, the biped doesn't stay on the chair very long and begins to leave the chair. Therefore, it is shown in the simulation that the biped cannot perform the sitting down motion by only tracking the angular positions. Moreover, increasing the feedback gains cannot help the tracking.

For the last simulation, the biped is not only tracking the angular positions but also the angular velocities. The feedback is then equal to:

$$- k_p(\Theta - \Theta_r) - k_v(\dot{\Theta} - \dot{\Theta}_r) \tag{24}$$

where Θ_r and $\dot{\Theta}_r$ are the angular position and angular velocities trajectories recorded in the experiment. Since during the motion of sitting down, the angular velocities change substantially, tracking the angular velocities seems to be required. Since it is proposed that the CNS can change the feedback gains for different movements, three different sets of feedback gains are chosen for the constrained phase. The position feedback gains are adjusted respectively to 1 time the minimum for stability(gain=1), 8 times the minimum for stability(gain=8) ,and 40 times the minimum for stability(gain=40). The velocity gains for all three cases are 0.1 times the value of the position gains. One important criteria to evaluate the feedback strategy is to examine the chair reaction force as shown in figure 8. In order to quantitatively differentiate which control strategy tracks the chair reaction force better, a root-mean-square(rms) error comparison is made between the curve in the simulation with the curve in the experiment. The rms error for gain=1 is 295.64, for gain=8 is 246.64 ,and for gain=40 is 106.63. From the rms criteria, the control strategy with higher feedback gains tracks the experimental results better. This is also true for the constraint which is shown in figure 9. Therefore, it is verified here that the CNS can change the feedback gain to better fit each movement.

From the above simulations, when the position feedback is set to one time the minimum gain required for stability, the feedback strategy that tracks the experimental angular positions and angular velocities shows the closest approximation to the experimental results. Physically, this means that the biped tracks both the angular positions and angular velocities when it is sitting down on a chair. Moreover, it is shown that when the position feedback gain for the third simulations is increased, there is a better approximation to the experimental results. This may mean that the CNS will change the feedback gains to give the human being more agility in daily motion.

V. Summary and Acknowledgement

It is shown that natural feedback supplied by muscles and modulated by the CNS is sufficient for stability. The feedback gains can be adjusted by the CNS for learned movements. The stability of this strategy for the uncontrained and the constrained motions is proven by the Lyapunov's second method. This control method is verified by computer simulations of sitting down motion. The computer simulations compare favorably with experimental results. The experimental results were supplied by Hichem Kallel and is gratefully acknowledged.

VI. References

[1] T. Bajd, A. Kralj, and R. Turk, "Standing-up of a Healthy Subject and a Paraplegic Patient," Journal of Biomechanics, Vol. 15, No. 1, page 1-10, 1982.

[2] A. Burstein, and V. Frankel, "The Viscoelastic Properties of Some Biological Materials," Annals of N.Y. Academy Sci., Vol. 146, pp. 158-165, 1968.

[3] C. Chow and D. Jocobson, "Studies of Human Locomotion via Optimal Programming," Mathematical Biosciences, Vol. 10, pp. 239-306, 1981.

[4] W. Chow, and E. Odell, "Deformations and Stresses in Soft Body Tissues of a Sitting Person," ASME Journal of Biomechanical Engineering, Vol. 100, pp. 79-87, May 1978.

[5] H. Hatze, "A Complete Set of Control Equations for the Human Musculo-Skeletal Sytem," Journal of Biomechanics, Vol. 10, pp. 799-805, 1977.

[6] H. Hemami, and V. Jaswa, "On a Three-Link Model of the Dynamics of Standing Up and Sitting Down," IEEE Transactions on Systems, Man and Cyernetics, Vol. SMC 8, No. 2, Feb. 1978.

[7] H. Hemami, and B. Wyman, "Modelling and Control of Constrained Dynamic Systems with Application to Biped Locomotion in the Frontal Plane," IEEE Transactions on Automatic Control, Vol. AC-24, No. 4, Aug. 1979.

[8] N. Hogan, "Stable Execution of Contact Tasks Using Impedance Control," IEEE International Conference on Research and Automation, pp. 1047-1054, 1987.

[9] H. Kallel, "A Six Link Planar Biped Model for Sitting Down on a Chair and Standing up," Master Thesis, Ohio State University, 1987.

[10] K. Lashley, "The accuracy of movement in the absence of excitation from the moving organ," American Journal of Physiology, Vol. 43, pp. 169-194, 1917.

[11] J. Laszlo, "Training of fast tapping with reduction of kinaesthetic, tactile, visual and auditory sensations," Quarterly Journal of Experimental Psychology, Vol. 19, pp. 344-349 1967.

[12] W. Levine, F. Zajac, M. Belzer, and M. Zomlefer, "Ankle Controls That Produce a Maximal Vertical Jump When Other Joints are Locked," IEEE Transactions on Automatic Control, Vol. AC-28, No. 11, Nov. 1983.

[13] N. McClamroch, and D. Wang, "Feedback Stabilization and Tracking of Constrained Robots," IEEE Transactions on Automatic Control, Vol. AC-33, No. 5, May 1988.

[14] L. Nashner and G. McCollum, "The Organization of Human Postural Movements: A Formal Basis and Experimental Synthesis," The Behavioral and Brain Sciences, Vol. 8, pp. 135-172, 1985.

[15] N. Reddy, H. Patel, G. Cochran, and J. Brunski, "Model Experiments to Study the Stress Distributions in a Seated Buttock," Journal of Biomechanics, Vol. 15, No. 7, pp. 493-504, 1982.

[16] L. Stark, "Neurological Control Systems," Plenum Press, New York, pp. 305, 1968.

[17] M. Takegaki, and S. Arimoto, "A New Feedback Method for Dynamic Control of Manipulators," ASME Journal of Dynamic Systems, Measurements, and Control, Vol. 102, June 1981.

[18] E. Taub and A. Berman, "Movement and learning in the absence of sensory feedback," in: The Neuropsychology of Spatially Oriented Behavior, ed. Freedman, pp. 173-192, 1968.

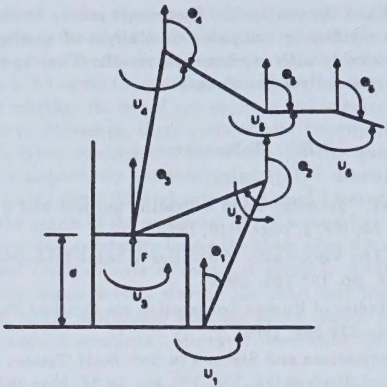

θ4
U4
θ5
θ6
U5
U6
θ3
θ2
U2
θ1
F
d
U3
U1

Figure 1: The biped

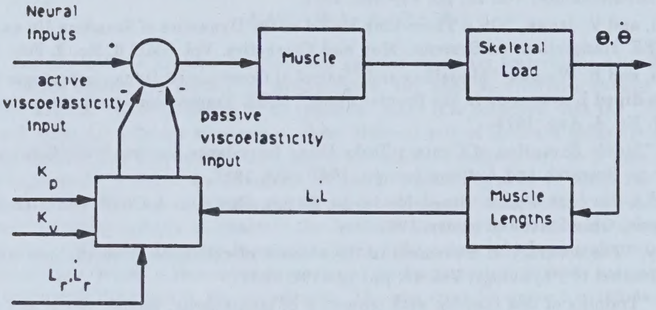

Neural inputs
+
active viscoelasticity input
-
-
passive viscoelasticity input
Muscle
Skeletal Load
θ, θ̇
K_D
K_V
L, L̇
Muscle Lengths
L_r, L̇_r

Figure 2: Dynamics at the Spinal Level

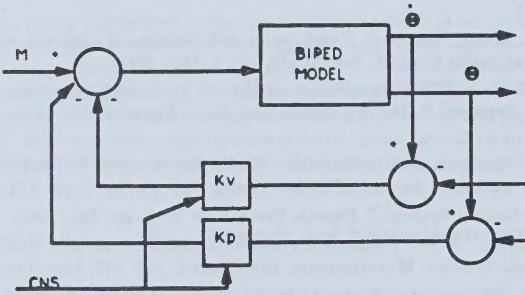

M
+
-
-
BIPED MODEL
θ̇
θ
Kv
Kp
+
-
θ̇_r
+
-
θ_r
CNS

Figure 3: The Block Diagram of the Motor Control System

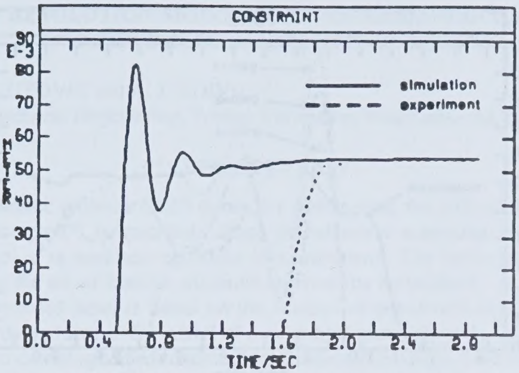


Figure 4: The Trajectory of Motion of the End-point of the Second Link in Point-to-Point Movement

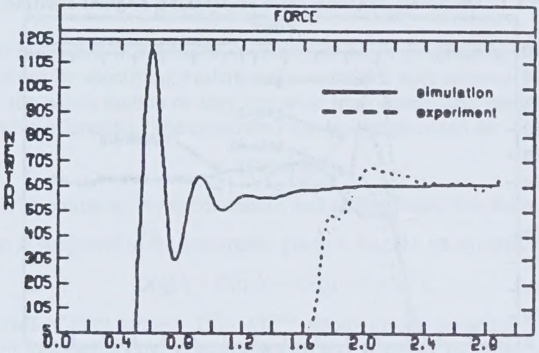


Figure 5: The Chair Reaction Forces in Point-to-Point Movement

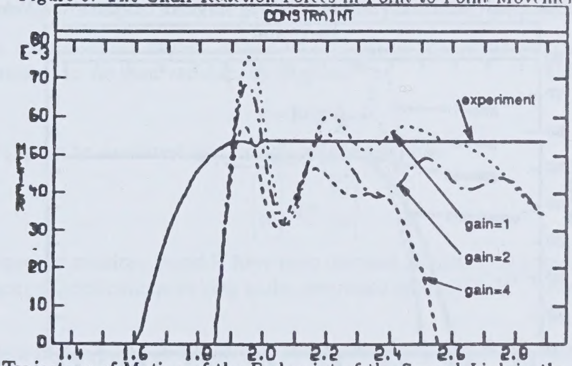


Figure 6: The Trajectory of Motion of the End-point of the Second Link in the Tracking Angular Positions

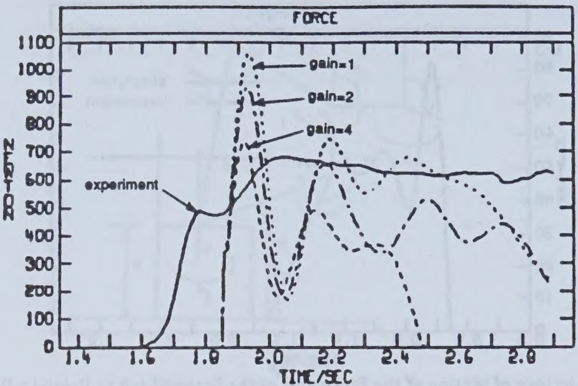


Figure 7: The Chair Reaction Force in Tracking Angular Positions

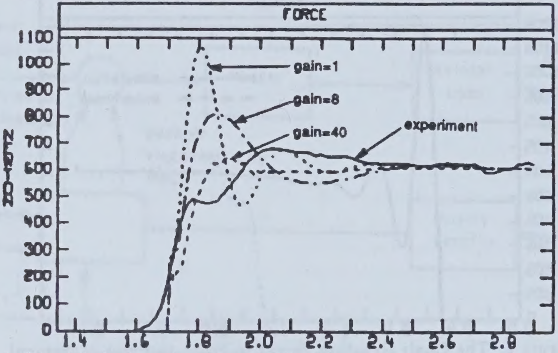


Figure 8: The Chair Reaction Force in Tracking Angular Positions and Velocities

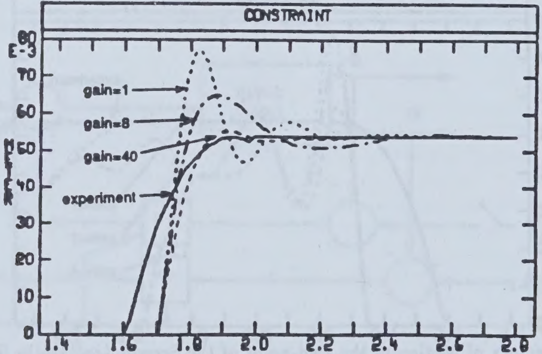


Figure 9: The Trajectory of Motion of the End-point of the Second Link in the Tracking Angular Positions and Velocities

REDUNDANCY RESOLUTION AND CONTROL OF REDUNDANT MANIPULATORS USING REDUCED-ORDER MODEL

S. H. ARNAUTOVIC and A. J. KOIVO
School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

INTRODUCTION

The inverse kinematic problem (IKP) stands for determining the joint space vector $q \in R^n$ when the task space vector $p \in R^m$ is specified. Since in redundant manipulators $m<n$, the IKP is an underspecified system of m nonlinear equations in n unknowns. The determination of the values of the joint variables in the set of feasible solutions resolves the redundancy. A method of the redundancy resolution, proposed here, is based on the concept of pseudovelocities,[1, 2]. By imposing general constraints, the reduced-order model of a redundant manipulator is obtained. The controller for the redundant arm can be specified in terms of the reduced order space vector.

REDUCED-ORDER MODEL

For an n-DOF redundant manipulator operating in an m-dimensional task space, $m < n$, the redundancy is resolved by specifying r additional constraints, with $n=m+r$. The constraints may be related directly to the actual motion or they can arise from optimizing the motion of the arm with respect to a specific cost function. The constraints can be characterized as:

$$H(q)\,\dot{q} = 0, \tag{1}$$

where $H(q) \in R^{r\times n}$ with entries h_{ij} being continuous and differentiable functions in all arguments.

When the robot is subjected to the constraints given in Eq. (1), its dynamical model is,[2]:

$$D(q)\,\ddot{q} + C(q,\dot{q}) + G(q) = \tau + H^T\lambda, \tag{2}$$

where $\tau \in R^n$ is vector of joint torques, $D(q) \in R^{n\times n}$ stands for the pseudoinertia matrix, $C(q,\dot{q}) \in R^n$ describes vector of centripetal and Coriolis forces and $G(q) \in R^n$ is a gravity vector. Vector λ represents r Lagrange's multipliers and the term $H^T\lambda$ is the generalized torque due to the constraints.

To derive a reduced-order model, vector $\nu \in R^m$, called the pseudovelocity vector, is introduced,[2]. It is related to the joint variables by $B(q) \in R^{m\times n}$:

$$\nu = B(q)\,\dot{q}. \tag{3}$$

Eqs. (1) and (3) can be combined into a single matrix equation:

$$\begin{bmatrix} B \\ H \end{bmatrix} \dot{q} = \begin{bmatrix} \nu \\ 0 \end{bmatrix} \tag{4}$$

where the arguments in matrices B and H have been dropped. Matrix B is chosen by the designer to satisfy certain design specifications as long as the composite matrix $[B^T\ H^T]^T$ has full rank n; hennce it is invertible.

It is convenient first to partition the inverse of the coefficient matrix into two matrices, an (nxm) matrix π_1, and an (nxr) matrix π_2 such that $B\pi_1=I_m$, $B\pi_2=0$, $H\pi_1=0$ and $H\pi_2=I_r$. Thus, it is possible to express the joint velocities and accelerations as:

$$\dot{q} = \pi_1\,\nu \tag{5}$$

$$\ddot{q} = \pi_1\,\dot{\nu} - (\pi_1\dot{B} + \pi_2\dot{H})\pi_1\nu. \tag{6}$$

To blend the constraint matrices H and B with the robot dynamics, $\dot{q}$ and $\ddot{q}$ are eliminated in Eq. (2). Premultiplying Eq. (2) by π_1^T leads to a set of m equations which enables one to explicitly solve for $\dot{\nu}$ since the coefficient matrix $(\pi_1^T D\pi_1)$ is invertible:

$$\dot{\nu} = (\pi_1^T D\pi_1)^{-1}\pi_1^T\,[\,D\,(\pi_1\dot{B} + \pi_2\dot{H})\pi_1\nu + \tau - C - G]. \tag{7}$$

Equation (7) represents an m-dimensional reduced-order dynamical model for an n-DOF manipulator when its motion is limited by r constraints, and $n=m+r$.

If B is chosen as a constant matrix, the pseudocoordinates by Eq.(3) are linear combinations of the joint positions. Since $\dot{B}=0$ (B is a constant matrix), the constrained model appears simplified. Alternatively, if $B=J_e$, where J_e is the Jacobian matrix of the manipulator, then the pseudovelocities become the task velocities $\dot{p}$. The variables operating in the reduced-order space are thus the task variables.

CONTROLLER DESIGN

The motion of the manipulator is described by the reduced-order dynamic model of Eq. (7). The output variables are $p(t)$ and $\dot{p}(t)$, while the input is the vector $\tau(t)$ of the generalized torques produced by the joint actuators. A desired trajectory is assumed to be specified in terms of the Cartesian task space variables p^d, $\dot{p}^d$ and $\ddot{p}^d$. The proposed controller will consist of 1) a primary controller $u^d(t)$ which makes a manipulator track the nominal trajectory under the ideal conditions, and 2) a secondary controller $\hat{u}(t)$ which compensates for small deviations from the nominal trajectory, caused by modeling errors and disturbances. The generalized torque will be then $\tau(t)=u^d(t)+\hat{u}(t)$.

The architecture for the primary controller $u^d(t)$ is selected according to Eq. (7) as:

$$u^d(t) = C(q^d,\dot{q}^d) + G(q^d) - D(q^d)[\pi_1(q^d)\dot{J}_e(q^d,\dot{q}^d) + \pi_2(q^d)\dot{H}(q^d,\dot{q}^d)]\pi_1(q^d)\dot{p}^d . \tag{8}$$

Equation (8) represents the inverse dynamics of the manipulator.

The secondary controller will be determined such that the task variables and the constraint forces $H^T\lambda$ are controlled independently. This goal is achieved by choosing $\hat{u}$ as:

$$\hat{u}(t) = \begin{bmatrix} H^T & D\pi_1 \end{bmatrix} \begin{bmatrix} \hat{u}_1(t) \\ \hat{u}_2(t) \end{bmatrix} \tag{9}$$

where the control inputs are $\hat{u}_1(t)\in R^r$ and $\hat{u}_2(t)\in R^m$.

The generalized torque $\tau(t)=u^d(t)+\hat{u}(t)$ is substituted into Eq. (7). Since $H\pi_1=0$, one obtains:

$$\ddot{p} = \hat{u}_2(t). \tag{10}$$

After obtaining the expression for λ from Eq. (2), the composite control $\tau=u^d+\hat{u}$ is applied. Assuming the perfect cancellation of the nonlinear terms, results in:

$$\lambda = -\hat{u}_1(t). \tag{11}$$

Equations (10) and (11) reveal that the expressions for the position and λ, i.e., the constraint form $H^T\lambda$ are decoupled and that they can be controlled independently.

As a conclusion, a method is proposed that resolves the redundancy giving rise to the reduced order manipulator model. The controller is specified in terms of the reduced order space vector. Moreover, the composite controller given by Eqs. (8) and (9) indeed decouple the task acceleration mode and the constraint forces in the Cartesian base coordinate system. Control inputs $\hat{u}_1$ and $\hat{u}_2$ can be realized by any linear control law (i.e, PI or PD law).

The described approach is illustrated by simulations of a three-link revolute planar manipulator with the constraints imposed on its motion in the form of Eq. (1).

REFERENCES

[1] A.J. Koivo, *Fundamentals for Control of Robot Manipulators,* J. Wiley & Sons, New York, NY, 1989.

[2] R. Kankaanranta and H.N. Koivo, Dynamics and Simulation of Compliant Motion of a Manipulator, *IEEE J. of Robotics and Automation* Vol. **RA-4, No 2** pp. 163-173, Apr 88.

A Modeling Mechanism for Hybrid Interacting Discrete-Event and Continuous-Time Systems

Weng H. Cheong † *: wcheong@eng.uiowa.edu*
Thomas L. Casavant : tomc@eng.uiowa.edu

Department of Electrical and Computer Engineering
University of Iowa
Iowa City, IA 52242

Abstract

The main purpose of this paper is to introduce a formal modeling technique, a hybrid modeling (HM) methodology, for **hybrid interacting discrete-event and continuous-time (HIDECT)** systems in which discrete-event (DE) subsystems exercise event-driven control over continuous-time (CT) subsystems. The paper presents a modeling technique that is an extension of the variable structure systems (VSS) approach and integrates two different modeling techniques into a HM methodology. In essence, the HM methodology encompasses Petri-Net (PN) based modeling technique [Rei83] to model the DE subsystem, and continuous-time simulation modeling technique [Hos88] to model the CT subsystem. DE-CT and CT-DE interfaces developed in the HM methodology bridge the two different subsystems. The ultimate goal of the HM methodology is to provide a basis for development of fundamental mathematical tools for analysis and control of HIDECT systems. To further this goal, the advantages and deficiencies of current mathematical tools are examined. In addition, properties of HIDECT systems are defined to serve as a benchmark for the evaluation of relevant mathematical tools. Finally, an example and its properties derived from the HM methodology are discussed.

1. Introduction

To date, there are ad-hoc modeling methodologies for the subarea of flexible manufacturing systems (FMS). However, there is no comprehensive modeling technique capable of representing HIDECT systems. In general, a modeling methodology for HIDECT systems should address the following questions. How does the DE system represent (possibly) complex control objectives of the HIDECT system? Can complex control objectives be defined in terms of primitive PN functional/decision blocks? What is the interface through which the DE subsystem exercises event-driven control over the CT subsystem? What information from the CT subsystem should be communicated to the DE domain? What is the interface through which the CT subsystem feeds back information to the DE subsystem? Is there a consistent mathematical representation for the CT subsystem, CT-DE interface, DE-CT interface, and DE subsystem?

Some of these questions are *partially* addressed in the application-specific area of FMS [MaS84, MaA86, MuK86]. In this context, suitability of use of an enhanced PN model for the control of an FMS is demonstrated in [KoK84]. However, developments in FMS's lack the fundamental tools that could be used for basic theoretical model development of HIDECT systems because all these developments make use of microprocessor-based systems to control devices in the FMS. The unavailability of these basic theoretical tools provides the motivation for the development of the HM methodology which is conducive to ease in formal mathematical analysis and design of HIDECT systems.

---

† Author's current address is Delco Electronics Corp., Kokomo, Indiana 46902.

This work was supported by the National Science Foundation under Grant Number ECS-8800910 and in part by CCR-8809600 and the NSF Software Engineering Research Center (SERC).

2. Concepts

As the nature of the CT and DE domains are inherently different, new concepts must be introduced to bridge the two domains so that information can be communicated between the two domains. Thus, the **region concept** is introduced to describe this finite subset of the n states in the CT domain which span an n-dimensional space. Each subspace of this subset is classified as a region which the event controller regards the crossing of a region boundary as a single event and each event is a part of the DE subsystem/event controller. The two main purposes of the region concept are to distinguish subspaces which have their own well-defined properties and to associate a set of possible actions to be taken when the state of the CT system enters a particular region.

Other concepts used in the HM methodology are explained here. First, the inputs and outputs of integrators of the plant of the CT subsystem are defined as *state variables*. Second, the *space of the CT subsystem* is the space spanned by the n state variables of the CT subsystem. Third, the *system state of the CT subsystem* is regarded as the instantaneous point in the space of the CT subsystem. The transition of the system state of the CT subsystem from one region to another signifies the occurrence of an event associated with that boundary. Fourth, an *active region* is referred to as the region which is associated with the active controller controlling the CT subsystem. Note that any point in space can lie in more than one region but it can lie only in one active region. Fifth, the collection of all the regions that are used to describe the space and all possible analog control configuration of the CT subsystem is referred as a *region status vector (RSV)*. Sixth, the process of mapping the space of the CT subsystem into the RSV is defined as *region mapping*. Seventh, a *decision-making (DM) entity* is the main component of a DE subsystem and is represented by a PN place in the HM methodology. A token in the place representing the DM entity indicates that an event corresponding to the DM entity has occurred. Eighth, the *complex control objectives* of the HIDECT subsystem are those control objectives that are not included in conventional control methodology, and are embodied within the structure of the PN.

3. Subsystems of HIDECT Systems

In the HM methodology, there are four subsystems of the HIDECT system — CT subsystem, DE subsystem, CT-DE interface, and DE-CT interface. Generally, the interfaces of the HIDECT system are used to couple the CT subsystem and DE subsystem into a feedback control system. Each subsystem performs a different function and the structure of each sub-system is different. In addition, each of the sub-systems has unique elements. The general structure of a HIDECT system and the interactions of its subsystems are shown in Figure 1.

3.1. CT Subsystem

As in conventional control methodology, the basic elements of the CT subsystem are integrators, summers, constant multipliers, and/or non-linear multipliers. The constant multipliers' values are defined as **parameters of the CT system**.

In conventional control methodology, the CT subsystem consists of only the plant as a single functional unit. However, in the HM methodology, the CT subsystem consists of the plant and zero or more analog controllers. The different input-output characteristics of the CT system can be obtained by a combination of the plant and different analog controllers. Hence the desired input-output characteristics can be obtained by activating certain analog controllers and deactivating all other analog controllers. This approach is similar to the variable structure control methodology, [Itk76, Utk77]. However, in HM methodology, two or more controller structures could be incorporated into the HIDECT system. At any time, there is only one active analog controller that is controlling the plant of the CT subsystem.

The capability to dynamically change analog controller structure is due to the use of constant multipliers which allow their values to be changed dynamically. The changes of the constant multiplier values are performed by the **analog generators** of the DE-CT interface.

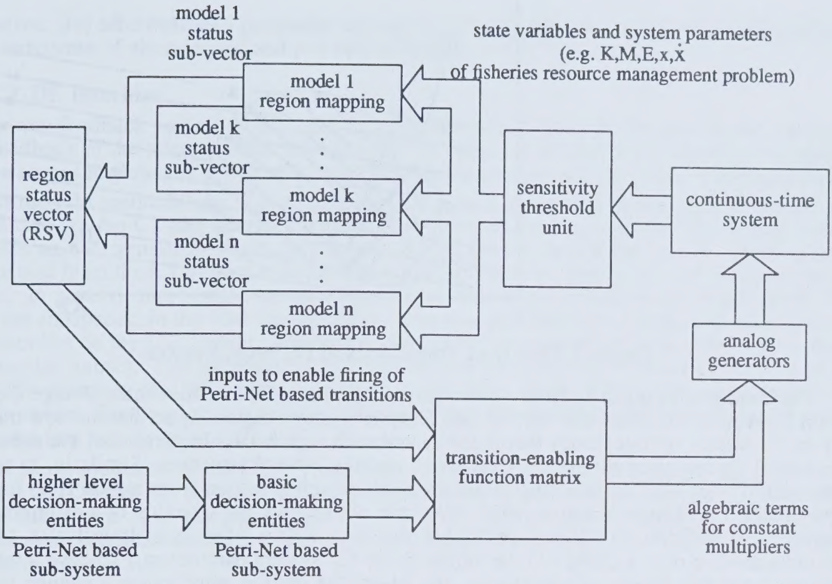

model 1
status
sub-vector
state variables and system parameters
(e.g. K,M,E,x,ẋ
of fisheries resource management problem)
model 1
region mapping
model k
status
sub-vector
region
status
vector
(RSV)
model k
region mapping
sensitivity
threshold
unit
continuous-time
system
model n
status
sub-vector
model n
region mapping
analog
generators
inputs to enable firing of
Petri-Net based transitions
transition-enabling
function matrix
higher level
decision-making
entities
basic
decision-making
entities
algebraic terms
for constant
multipliers
Petri-Net based
sub-system
Petri-Net based
sub-system

Figure 1. General Structure of HIDECT System

3.2. DE Subsystem

The DE system consists of an RSV, which represents the various sets of possible regions spanning the space of the CT system, and DM entities. In the HM methodology, the DE elements (regions and DM entities) and their interactions are represented as PN places and transition-enabling functions (TEFs) in an analogous manner to that of the normal PN modeling techniques [Rei83]. In addition, the DM entities have a two-level hierarchical structure which allows simple control objectives to be implemented by basic DM entities and complex control objectives to be implemented by higher-level DM entities which influence the basic DM entities. The interconnection of DM entities reveals their interdependencies and it is assumed that these interdependencies have been obtained through a decision analysis as described in [HoM83] or have evolved through learning in a production system [New73].

The simple control objectives are related to the control criteria that are used in the design of an analog controller in the conventional control methodology and the functions of the basic DM entities, which implement the simple control objectives, are (i) maintain/change the active region in which the system state of the CT subsystem lies, (ii) set type of input - e.g., types 0, 1, or 2, (iii) set type of control structure - e.g., types 0, 1, or 2, (iv) set amount of overshoot, M_0, in percentage, (v) set desired undamped natural frequency in radians, (vi) set position of poles of the CT subsystem - e.g., using pole placement control strategy, and (vii) set arrangement of controller - e.g., cascade, feedback, or cascade + feedback.

The active region corresponds to a combination of the plant and a specific analog controller. Changing from one active region to another may involve the same analog controller if the two regions are associated with the same analog controller.

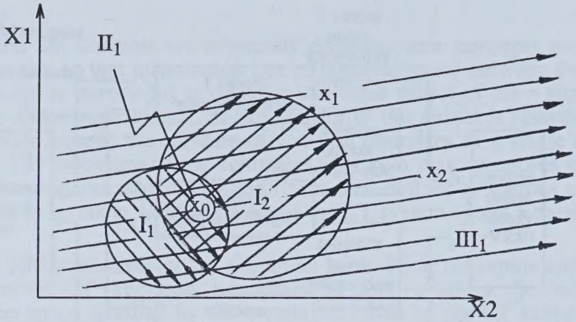


Figure 2. Family of Trajectories of Different Regions

For example, to reach x_2 from x_0 in Figure 2, the basic DM entities must change the active region from I_1 to III_1 when the state of the CT system enters region II_1 so that the new trajectory is from the family of trajectories that is associated with region III_1. In particular, the subscript of the symbol representing the region indicates a specific control structure. Similarly, to reach x_1 from x_0, the basic DM entities must cause a change in active region from region I_1 to region I_2. In particular, the change of active region may/may not involve the selection of a different analog controller. In addition, to follow a particular trajectory within a family of trajectories, the basic DM must involve only a change to the values of the CT system parameters. To have a trajectory different from this family of trajectories, the basic DM entities must cause a change in active region so that a different family of trajectories is available. In other words, a different analog control structure must be selected. In general, the structure of the interconnection within the level of basic DM entities is linear in the sense that all of the basic DM entities are inputs to the DE-CT interface. It is also of a deterministic structure.

The type of input is defined as the highest degree of the input to the controller. For example, a step input ($u(t) = a$, where a is a constant) is considered a type 0 input. Similarly, input of the form $u(t) = at$ is regarded as a type 1 input, and $u(t) = at^2$ is a type 2 input.

The type of control structure is defined as the degree of the pole at the origin in the following equivalent transfer function.

$$G_e(s) = \frac{K_v(1 + b_1 s + b_2 s^2 + \cdots + b_m s^m)}{s^i(1 + a_1 s + a_2 s^2 + \cdots + b_{n-i} s^{n-i})} = \frac{G_{tot}(s)}{1 - G_{tot}(s)}$$

where
$G_e(s)$ = equivalent transfer function
G_{tot} = total transfer function of the control system
K_v = gain factor
i = type of control structure/system

The relationship between type of input and type of control system with regard to steady-state error is described in [Oga87]. In the HM methodology, the type of control system can be changed dynamically in response to a change in type of input so that the steady-state error is determinable. The various input and control system types are represented by basic DM entities of the DE subsystem.

In addition to the simple control objectives, the DE subsystem/event-driven controller also implements the complex control objectives required in a desired "intelligent" HIDECT control system so that the system can function predictably under complex environmental conditions. Such complex control objectives [Hol89] may include: (i) **preferences** - e.g., state preferences, (ii) **constraints** - hard and soft constraints, (iii) **decision variables** - decision elements of an alternative, (iv) **alternatives** - particular settings of several decision variables, (v) **uncertainties** about **outcomes** of alternatives, and (vi) **risk attitude**.

3.3. CT-DE Interface

In any feedback control system, the states of the system must be fed back to the controller. This feedback of the states enables the controller to make corrections to the inputs to the plant so that the output response of the plant follows a desired trajectory and the states reach equilibrium.

In the HM methodology, the system state of the CT subsystem is fed back to the DE subsystem through the CT-DE interface which implements the region concept. The region concept describes an n-dimensional space spanned by the CT system as a finite subset. Thus, essential information from the CT system may be communicated back to, and be able to influence, the DE system. In general, there could be more than one subset which describes the same space. Each set of the subspaces, in the HM methodology, selects a particular combination of plant and analog controller in the CT system. Therefore, a region of one subset may overlap with a region from another subset. The collection of all these subsets forms the RSV which is represented by a set of PN places in the DE system model. Each element of the RSV represents a region of a specific subset. Each region is a subspace enclosed by a set of switching hyperplanes (or switching planes in a two dimensional space).

In short, elements of the CT-DE interface implement the region concept which is used in bridging the CT and DE domains so that information from the CT subsystem can be fed back to the DE subsystem. The main elements of the CT-DE interface are a **sensitivity threshold unit** and **parallel computation units** which perform the region mapping of the system state of the CT subsystem into the RSV.

3.4. DE-CT Interface

The main components of the DE-CT interface are **analog generators** which take inputs from a sub-set of PN places of the PN-modeled DE subsystem and convert it into analog values for the parameters of the CT system. Basically, the DE subsystem/event-driven controller controls the CT subsystem through the DE-CT interface so that the complex control objectives, the desired input-output characteristics and/or equilibrium state and/or desired trajectory are satisfied. The other component of the DE-CT interface are PN-modeled TEFs which encode the various conditions for setting particular parameter values of the analog controller of the CT subsystem. The outputs of the TEFs are the inputs to the various analog generators.

4. Current Mathematical Tools

The HM methodology may be viewed as an extension and combination of four current mathematical tools. The first tool examined is the variable structure design methodology [Itk76, Utk77] from the control engineering area. The next two mathematical tools come from the computer engineering area of DE system - — Murata's incidence matrix [Mur77] and Da's reflexive incidence matrix [DaA87]. Last, the theory of multivariable polynomial matrices [Kri85] is considered for possible applicability to model HIDECT system in terms of control laws having multivariable polynomial matrices.

5. Definitions of Properties of HIDECT Systems

Before a hybrid tool is proposed, properties of HIDECT system are defined so that results derived from any new mathematical tool could be expressed in terms of these properties.

Definition 1: The state of a HIDECT system is defined as follow:

$$(x_1[k+1], \ldots\ldots, x_n[k+1]) = f(x_1[k], \ldots\ldots, x_n[k], \dot{x}_1[k], \ldots., \dot{x}_n[k], d_1, \ldots., d_p)$$

where

$x_i[k]$ = i^{th} state variable at "time" instance k,
$d_1, \ldots, d_p \in D$ and is invariant with respect to k, with
D = decision domain of DE subsystem.

Definition 2: The state $(x_1, ..., x_n)$ of a HIDECT system is said to be unique when

$(x_1[k], ..., x_n[k], \dot{x}_1[k], ..., \dot{x}_n[k]) \rightarrow (x_1[k+1], ..., x_n[k+1])$

where

$(x_1[k+1], ..., x_n[k+1])$ can only be reached by making decisions $(d_i, ..., d_j)$ simultaneously

Definition 3: An essential (critical) precondition of a state of a HIDECT system occurs when

$f: (x_1[k], ..., x_n[k], \dot{x}_1[k], ..., \dot{x}_n[k]) \rightarrow (x_1[k+1], ..., x_n[k+1])$

where

$f(d_r, d_s, d_t) = f(d_p, d_s)$, and

d_s = essential precondition of $(x_1[k+1], ..., x_n[k+1])$, and

d_p, d_r, and d_t = non-essential preconditions of a particular system state.

Definition 4: A non-essential precondition of a state of a HIDECT system occurs when

$f(d_r, d_s, d_t) = f(d_s)$, and

d_s = essential precondition of $(x_1, ..., x_n)$, and

d_r, and d_t = non-essential preconditions of a particular system state.

In other words, the system state possesses disturbance rejection capabilities with regard to decisions, d_r and d_t.

Definition 5: Reachability of a state of a HIDECT system can be attained only if all essential preconditions of the system state are satisfied

Definition 6: Minimum energy of a state of a HIDECT system occurs when the set of decisions made to reach the system state is a minimum set.

Definition 7: The range of a state of a HIDECT system is defined as the space of reachable system states spanned by all possible decision sets.

Definition 8: A domain of a state of a HIDECT system is defined as space spanned by possible originating states of the system.

Definition 9: A decision set of a HIDECT system is said to be robust when the decision set acts on only one specific system state and it results in only one system state.

Defintion 10: A HIDECT system is well posed only when all system states are unique.

Definition 11: Deadlock occurs when there are one or more self-looped decision components.

6. Example

With the properties of HIDECT system as defined above, an example dealing with management of a fisheries resource is given to demonstrate the use of the HM methodology. In this example [McC80, PeD88], a simple model of a single species of fish is developed for the purpose of illustrating some of the dynamic capabilities of the HM methodology. In this fisheries model, the fish is a renewable resource as it has a natural reproduction rate. The management problem is to harvest or maintain this resource according to some (possibly complex) high-level policy objectives.

In the HM methodology, the dynamics of the fish population is considered as that of a plant of the CT subsystem which has a growth rate accordingly to the equation,

$$\dot{x} = Kx(M - x) - u \qquad (1.1)$$

where

x = fish population,

K, M = positive constants for natural rate of growth, and maximum pond population, respectively, and

u = harvesting rate or input from the analog controller of the CT subsystem.

For this example, we assume that there are three simple control objectives or harvesting policies. The three different harvesting policies are "no harvesting" (NH), "constant harvesting" (CH), and "constant effort harvesting" (CEH).

The three different harvesting policies and plant may be represented by the equation,

$$\dot{x} = Kx(M - x) - (H + Ex) \qquad (1.2)$$

where
H = constant harvesting rate; a constant multiplier,
Ex = constant effort harvesting rate; E, a constant multiplier,
x, $\dot{x}$ = state variables, and
K, M = system parameters.

With regard to the three control policies, a short summary of the characteristics of each is described here. With the NH policy, the trajectory in which the system state moves is restricted to the path described by equation 1.1 with $u = 0$. Thus, the NH policy offers only one stable equilibrium state, and to have a stable system state that is different from (M,0), a different control structure/ harvesting policy is required. For the CH policy, the CT system is unable to maintain the fish population, x, at a stable system state when the maximum harvesting rate of $\frac{KM^2}{4}$ is used. In general, the trajectory in which the system state moves is restricted to the path described by equation 1.2 with $E = 0$ and $H \geqslant 0$. When a stable system state is desired that is not in the range between $(\frac{M}{2}, 0)$ and (M, 0) or a harvesting rate greater than $\frac{KM^2}{4}$, a different control structure/harvesting policy is required. Last, the CEH policy provides a wide range of stable system states from (0, 0) to (M, 0) and a maximum possible harvesting rate of $\frac{KM^2}{4}$ which leads to an asymptotically stable system state. The trajectory in which the system state moves is restricted to the path described by equation (1.2) with $E \geqslant 0$ and $H = 0$. When a stable system state that is beyond (M,0) is desired, a different control structure/harvesting policy is required.

The general structure of a HIDECT system for the fisheries resource management problem is shown in Figure 3. The influence of the CT subsystem on the DE subsystem through the CT-DE interface and the influence of the DE subsystem on the CT subsystem through the DE-CT interface are also shown in the same figure.

An example, as shown in Figure 4, is used to illustrate the dynamic capabilities of the HM methodology as applied to the fisheries resource management problem. For instance, the desired complex control objectives may be a preference for the sequence of $\{x_0 \rightarrow x_a \rightarrow x_b\}$, where $\frac{M}{2} \leqslant x_a \leqslant M$, $0 \leqslant x_b \leqslant \frac{M}{2}$, and a preference of using the CH policy over the CEH policy. To accomplish these complex control objectives, the higher-level DM entities will activate the basic DM entity, which identifies the preference of CH policy over the CEH policy, and the basic DM entity corresponding to the selection of a CH policy. The DE-CT interface will then set $H = K[\frac{M^2}{4} - (x_a - \frac{M}{2})^2]$ because x_0 is in region II_1 of Figure 4. After the system state reaches stable state $(x_a, 0)$, the basic DM will switch the control structure to that of the CEH policy. That is, because x_a lies in region I_2 of Figure 4, the following action will be taken: set $E = K(M - x_b)$. The path in which the system state moves to accomplish these complex control objectives is shown in Figure 4.

Thus, it can be observed that the HM methodology enables complex control objectives to be accomplished and the accommodation of more than one control structure. As more control structures are added to the HIDECT systems, greater dynamic capabilities will be incorporated in terms of a greater combinations of actions being available.

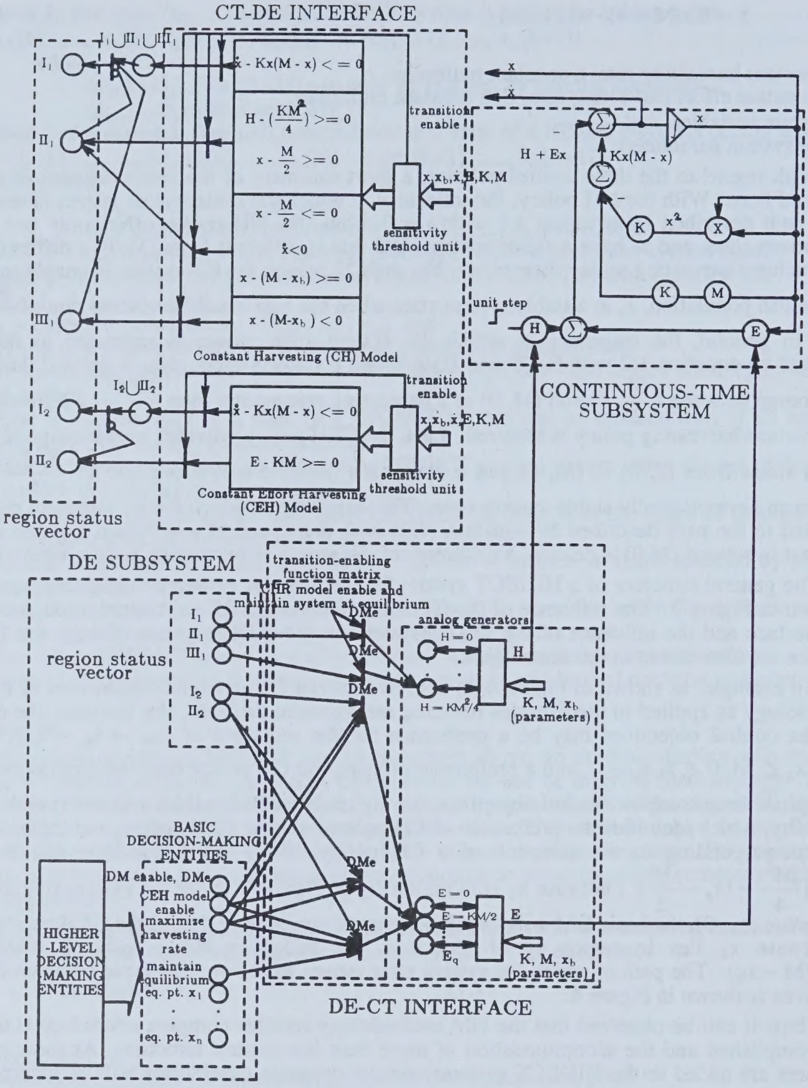

CT-DE INTERFACE
$I_1 \cup II_1 \cup III_1$
I_1
II_1
III_1
$\dot{x} - Kx(M - x) <= 0$
$H - (\frac{KM^2}{4}) >= 0$
$x - \frac{M}{2} >= 0$
$x - \frac{M}{2} <= 0$
$\dot{x}<0$
$x - (M - x_b) >= 0$
$x - (M-x_b) < 0$
Constant Harvesting (CH) Model
transition enable
$x, x_b, \dot{x}, H, K, M$
sensitivity threshold unit
$I_2 \cup II_2$
I_2
II_2
$\dot{x} - Kx(M - x) <= 0$
$E - KM >= 0$
transition enable
$x, x_b, \dot{x}, E, K, M$
sensitivity threshold unit
Constant Effort Harvesting (CEH) Model
region status vector
x
$\dot{x}$
$-$
$H + Ex$
$Kx(M - x)$
x^2
K
X
K
M
unit step
H
E
CONTINUOUS-TIME SUBSYSTEM
DE SUBSYSTEM
transition-enabling function matrix
CHR model enable and maintain system at equilibrium
region status vector
I_1
II_1
III_1
I_2
II_2
DMe
DMe
DMe
DMe
DMe
analog generators
$H = 0$
$H = KM^2/4$
H
K, M, x_b (parameters)
BASIC DECISION-MAKING ENTITIES
DM enable, DMe
CEH model enable
maximize harvesting rate
maintain equilibrium
eq. pt. x_1
eq. pt. x_n
HIGHER -LEVEL DECISION MAKING ENTITIES
$E = 0$
$E = KM/2$
E_q
E
K, M, x_b (parameters)
DE-CT INTERFACE

Figure 3. General Structure of HIDECT System
for the Fisheries Resource Management Problem

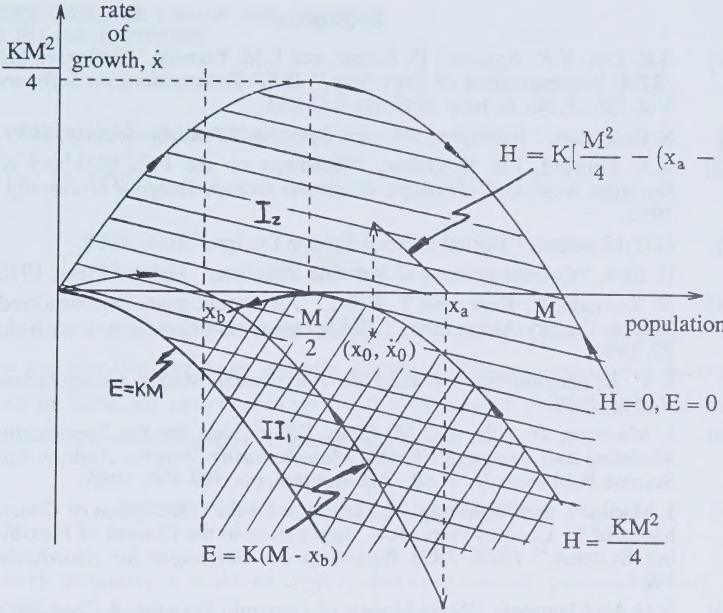


Figure 4. Example of the Dynamic Capabilities of HM Methodology

7. Conclusions

The HM methodology for formal modeling of HIDECT systems is based mainly on mappings of the space of the CT subsystem onto that of the DE subsystem and vice versa so that the two subsystems can be mapped to a common representation. In this work, we have chosen the Petri-Net (PN) representation. The HM methodology provides flexibility in terms of dynamic selection of control structures. Also, HIDECT systems designed based on this HM methodology offer greater adaptability in terms of more general applicability to a larger domain than that of CT subsystems. This larger domain includes domains of both DE and CT subsystems. Furthermore, the advantages of the HM methodology provide the motivation for mathematical analysis of the HIDECT system. In essence, the mathematical representation of the HIDECT system must include region-mapping and mapping of decision-making onto control structure. The mathematical analysis of HIDECT systems, as well as details concerning PN modeling of higher-level DM entities is beyond the scope of this paper, and is the subject of on-going research.

8. Acknowledgements

The authors gratefully acknowledge many useful discussions with R.A. DeCarlo in the early phases of this work.

References

[DaA87] S.K. Das, V.K. Agrawal, D. Sarkar, and L.M. Patnaik, ''Reflexive Incidence Matrix (RIM) Representation of Petri Nets,'' *IEEE Transactions on Software Engineering,* Vol. SE-13, No. 6, June 1987, pp. 643-653.

[Hol89] S. Holtzman, ''Intelligent Decision Systems,'' Addison-Wesley, 1989.

[HoM83] R.A. Howard, J.E. Matheson, ''Readings on the Principles and Applications of Decision Analysis,'' *Strategic Decisions Group, Stanford University,* Vol. I and II, 1983.

[Hos88] G.H. Hostetter, ''Digital Control System Design,'' Holt, 1988.

[Itk76] U. Itkis, ''Control systems of Variable Structure,'' Halsted Press, 1976.

[KoK84] N. Komoda, K. Kera, and T. Kubo, ''An Autonomous Decentralized Control System for Factory Automation,'' *IEEE Transactions on Computer,* Vol. C-17, pp. 73-83, 1984.

[Kri85] E.V. Krishnamurthy, ''Error-Free Polynomial Matrix Computations,'' Springer-Verlag, 1985.

[MaA86] J. Martinez, H. Alla, and M. Silva, ''Petri Nets for the Specification of FMSs,'' *Modeling and Design of Flexible Manufacturing Systems,* Andrew Kusiak, Elservier Science Publishers B. V. ed., Armsterdam, pp. 389-406, 1986.

[MaS84] J. Martinez, and M. Silva, ''A Language for the Description of Concurrent Systems Modeled by Colored Petri Nets: Application to the Control of Flexible Manufacturing Systems,'' *IEEE 1984 Workshop on Languages for Automation,* pp. 72-77, 1984.

[McC80] N.H. McClamroch, ''State Models of Dynamic Systems: A Case Study Approach,'' Springer-Verlag, 1980, pp. 99-104.

[MuK86] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna, ''A Petri Net-Based Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation,'' *IEEE Transactions on Industrial Electronics,* Vol. IE-33, No. 1, pp. 1-8, Feb. 1986.

[Mur77] T. Murata, ''State Equation, Controllability, and Maximal Matchings of Petri Nets,'' *IEEE Trans. on Automatic Control,* June 1977, pp. 412-415.

[New73] A. Newell, ''Production Systems: Models of Control Structures,'' *Visual Information Processing,* W.G. Chase (ed.), Academic Press, 1973.

[Oga87] ''Discrete-Time Control Systems,'' Prentice-Hall, 1987.

[PeD88] P. Peleties, R.A. DeCarlo, ''Modeling of Interacting Continuous Time Discrete Event Systems: An Example,'' *Proceedings 26th Allerton Conference,* Sep. 1988, pp. 1150-1159.

[Rei83] W. Reisig, ''Petri Nets, An Introduction,'' Springer-Verlag, New York, 1983.

[Utk77] V.I. Utkin, ''Variable Structure Systems with Sliding Modes,'' *IEEE Trans. on Automatic Control,* Vol. AC-22, No. 2, Apr. 1977, pp. 212-222.

REDUCED ORDER OBSERVER DESIGN FOR DISCRETE STOCHASTIC BILINEAR SYSTEMS

ENGIN YAZ
Electrical Engineering Department
University of Arkansas
Fayetteville, AR 72701

ABSTRACT

We consider a general discrete time stochastic bilinear system description and derive the mean square optimal, linear, unbiased observer equations to be used in constructing a certain linear combination of state variables based on noisy output measurements.

INTRODUCTION AND PROBLEM STATEMENT

This work presents a reduced order observer design procedure for system and measurement models which contain multiplicative as well as additive noise components. Such models are alternatively called stochastic bilinear, state-dependent noise models or multiplicative noise models. Due to the several application areas such as satellite attitude control [1], chemical reactor control [2], macroeconomics [3], population dynamics [4], time-sharing and random round-off errors in computer operation [5], and recently, robustness studies [6]-[8], estimators have been constructed [9]-[14] for various models. In this study, we consider the general stochastic bilinear model used in the works [12]-[14] to extend the mean square optimal, unbiased reduced order observer results of [15] for deterministic parameter systems to this case.

Let us consider the discrete stochastic bilinear system

$$x_{k+1} = A_k(\omega)x_k + v_k(\omega) \tag{1}$$

with the measurement equation

$$y_k = C_k(\omega)x_k + w_k(\omega)$$

where $x_k \epsilon \mathbb{R}^n$ is the state, $y_k \epsilon \mathbb{R}^m$ the output, $v_k \epsilon \mathbb{R}^n$ is the white additive process noise of zero mean (denoted by $E\{v_k\} = \bar{v}_k = \bar{v} = 0$) and covariance V, and $w_k \epsilon \mathbb{R}^m$ is the white measurement noise with zero mean and covariance W, A_k and C_k are weakly stationary, stochastic matrix sequences with nonzero means, whose elements are white random sequences. The initial state x_o is of mean $\bar{x}_o$, covariance P_o and is independent of other noise sources. A_k, C_k, v_k, w_k are assumed to be mutually uncorrelated second order processes. The problem is to reconstruct $\hat{z}_k$ the estimate of $z_k \epsilon \mathbb{R}^\ell$ which may be of lower dimension ($\ell \leq n$) than x_k and is given by

$$z_k = Lx_k \tag{3}$$

MAIN RESULTS

To accomplish the above aim, we will use a linear observer (due to its computational simplicity)

$$\hat{z}_{k+1} = F_k\, z_k + K_{k+1}\, y_{k+1} \tag{4}$$

First, by defining the observation error

$$e_k = z_k - \hat{z}_k \tag{5}$$

we need to determine F_k and the observer gain sequence K_{k+1}: i) to guarantee unbiased estimation ($E\{e_k\} = 0$, for all $k \geq 0$) ii) to obtain a mean-square optimal observer, i.e., the criterion

$$J = E\{e_N^T S e_N + \sum_{k=0}^{N-1} e_k^T\, U_k e_k\} \tag{6}$$

where S and U_k are symmetric and nonnegative definite, is to be minimized. The observation error e_k evolves in time as follows:

$$e_{k+1}=F_k e_k+(LA_k-K_{k+1}C_{k+1}A_k-F_kL)x_k+(L-K_{k+1}C_{k+1})v_k-K_{k+1}w_{k+1} \tag{7}$$

To satisfy the unbiasedness requirement, we let $\hat{z}_0 = L\bar{x}_0$ and

$$L\bar{A} - K_{k+1}\,\bar{C}\,\bar{A} - F_kL = 0$$

or

$$\psi = L\,\bar{A} = [F_k \;\; K_{k+1}] \begin{bmatrix} L \\ \bar{C}\;\;\bar{A} \end{bmatrix} \tag{8}$$

As commented in [15] before for deterministic-parameter case, this condition will be satisfied if

$$\text{rank} \begin{bmatrix} L \\ \bar{C}\;\bar{A} \end{bmatrix} = n \tag{9}$$

Let us now choose the n independent rows of $\begin{bmatrix} L \\ \bar{C}\;\bar{A} \end{bmatrix}$ and stack them to form the matrix H_1 and let the remaining rows be combined to form H_2. The n columns of $[F_k \;\; K_{k+1}]$ corresponding to H_1 in (8) are called G_1 and the others are called G_2. If

$$H_1 = S_1 \begin{bmatrix} L \\ \bar{C}\;\bar{A} \end{bmatrix} \tag{10}$$

where $S_1 \in \mathbb{R}^{n\times(m+\ell)}$ is an orthogonal matrix ($S_1S_1^T = I_n$) with elements ones and zeros, we have

$$G_{1k} = [F_k \;\; K_{k+1}]\; S_1^T \tag{11}$$

Since (8) can be rewritten as

$$G_{1k}H_1 + G_{2k}H_2 = \psi \tag{12}$$

we have

$$G_{1k} = (\psi - G_{2k}H_2)H_1^{-1} \tag{13}$$

using the invertibility of H_1. Now, we can write F_k and K_{k+1} in terms of G_{1k} and G_{2k} by proper choice of matrices M_1, M_2, N_1 and N_2 which again contain only ones and zeros:

$$F_k = G_{1k}M_1 + G_{2k}M_2, \quad K_{k+1} = G_{1k}N_1 + G_{2k}N_2 \tag{14}$$

Combining (13) and (14), we get

$$F_k = G_{2k}\,\Gamma_2 + \Gamma_1, \quad K_{k+1} = G_{2k}\,\Omega_2 + \Omega_1 \tag{15}$$

with

$$\Gamma_1 = \psi H_1^{-1}M_1, \quad \Gamma_2 = M_2 - H_2\,H_1^{-1}M_1 \tag{16a}$$

$$\Omega_1 = \psi\,H_1^{-1}\,N_1, \quad \Omega_2 = N_2 - H_2\,H_1^{-1}\,N_1 \tag{16b}$$

Using these new variables, equation (7), condition (8), introducing $A_k = \bar{A} + \tilde{A}_k$, $C_k = \bar{C} + \tilde{C}_k$, and after some lengthy manipulations, the evolution of the observation error covariance $E\{e_k e_k^T\} \overset{\Delta}{=} P_k^e$ is obtained as

$$P_{k+1}^e = (G_{2k}\Gamma_2+\Gamma_1)P_k^e(G_{2k}\Gamma_2 + \Gamma_1)^T + (G_k\Omega_2+\Omega_1)\Phi_{1k}(G_{2k}\Omega_2+\Omega_1)^T$$

$$+ L\Phi_{2k}L^T - L\Phi_{2k}\bar{C}^T(G_{2k}\Omega_2+\Omega_1)^T - (G_{2k}\Omega_2+\Omega_1)\,\bar{C}\,\Phi_{2k}L^T \tag{17}$$

$$\Phi_{1k} = W + \overline{CVC^T} + \overline{\tilde{C}\,\overline{\tilde{A}X_k}\tilde{A}^T\tilde{C}^T} + \overline{\tilde{C}AX_k\bar{A}^T\tilde{C}^T} + \overline{\bar{\tilde{C}}AX_k\tilde{A}^T\bar{C}^T} \tag{18}$$

$$\Phi_{2k} = \overline{\tilde{A}X_k\tilde{A}^T} + V \tag{19}$$

and

$$X_k = E\{x_k x_k^T\} \tag{20}$$

which satisfies

$$X_{k+1} = \overline{AX_kA^T} + V, \quad X_o = \bar{x}_o\bar{x}_o^T + P_o \tag{21}$$

Expressing the cost functional (6) using the interchangeability of trace and expectation operations, as

$$J = tr[S\,P_N^e + \sum_{k=0}^{N-1} U_kP_k^e] \tag{22}$$

we form the Hamiltonian

$$H = U_kP_k^e + P_{k+1}^e\,\Lambda_{k+1}^T \tag{23}$$

Λ_k being the Lagrange multiplier matrix. Substituting from (17) and performing $(\partial H/\partial G_{2k})=0$, one obtains

$$G_{2k} = -(\Gamma_2P_k^e\Gamma_1^T + H_k)D_k \tag{24}$$

where

$$H_k = \Omega_2\Phi_{1k}\Omega_1^T - \Omega_2\bar{C}\Phi_{2k}L^T, \quad D_k=(\Gamma_2P_k^e\Gamma_2^T+\Omega_2\Phi_{1k}\Omega_2^T)^{-1} \tag{25}$$

It is interesting to note that the optimal gain is independent of the weighting matrices in (22). So, from (15) we have

$$F_k = \Gamma_1 - (\Gamma_2 P_k^e \Gamma_1^T + H_k) D_k \Gamma_2 \tag{26}$$

$$K_{K+1} = \Omega_1 - (\Gamma_2 P_k^e \Gamma_1^T + H_k) D_k \Omega_2 \tag{27}$$

and when G_{2k} from (24) is substituted into (17), we obtain

$$P_{k+1}^e = \Gamma_1 P_k^e \Gamma_1^T - (\Gamma_1 P_k^e \Gamma_2^T + H_k^T) D_k (\Gamma_1 P_k^e \Gamma_2^T + H_k^T)^T$$

$$+ [\Omega_1, -L] \begin{bmatrix} \Phi_{1k} & \overline{C}\Phi_{2k} \\ \Phi_{2k}\overline{C}^T & \Phi_{2k} \end{bmatrix} \begin{bmatrix} \Omega_1^T \\ -L^T \end{bmatrix} \tag{28}$$

together with the initial condition $P_0^e = LP_0L^T$, which completes our derivation of the mean square optimal unbiased reduced order observer.

CONCLUSION

We have derived the mean square optimal unbiased linear observer equations for discrete-time stochastic bilinear systems. We remark here that both F_k and K_{k+1} to be used in the observer (4) implementation can be computed beforehand using (16a,b), (18)-(19), (21), (25)-(28) and stored in memory. One can see that although the observer Riccati equation (28) is reduced in dimension, we still need to iterate (21) which is an nxn matrix difference equation as in the full-order case [6].

Another similarity to the full-order case is the necessity , in the steady state case, of the mean square boundedness of the system state to guarantee the convergence of X_k in (21) to a constant value. The condition $\rho(\overline{A \otimes A}) < 1$ where ρ is the spectral radius and $\otimes$ the Kronecker product secures the mean square boundedness of the state.

REFERENCES

[1] McLane, P. J., "Optimal stochastic control of linear systems with state- and control-dependent disturbances," IEEE Trans. Autom. Cont., 16, 793, 1971.

[2] Rao, N. J., D. Ramakrishna, and D. J. Borwanker, "Nonlinear stochastic simulation of stirred tank reactors," Chem. Engin. Sci., 29, 1193, 1974.

[3] Aoki, M., Optimal Control and System Theory in Dynamic Economic Analysis, Elsevier, New York, 1976.

[4] Mohler, R. R., and W. J. Kolodziej, "An overview of stochastic bilinear control processes," IEEE Trans. Syst. Man Cyber., 10, 913, 1980.

[5] Van Wingerden, A. J. M. and W. L. DeKoning, "The influence of finite word length on digital optimal control," IEEE Trans. Autom. Cont., 29, 385, 1984.

[6] Willems, J. L. and J. C. Willems, "Robust stabilization of uncertain systems," SIAM J. Optim. Cont., 21, 352, 1983.

[7] Bernstein, D. S. and D. C. Hyland, "The optimal projection/maximum entropy approach to designing low-order, robust controllers for flexible structures," Proc. of 24th CDC, Ft. Lauderdale, FL, 745, 1985.

[8] Yaz, E., "Deterministic and stochastic robustness measures for discrete systems," IEEE Trans. Autom. Cont., 33, 952, 1988.

[9] Nahi, N.E., "Optimal recursive estimation with uncertain observation," Trans. Inform. Theory, 15, 457, 1969.

[10] Rajasekaran, P. K., N. Satyanarayana, and M. D. Srinath, "Optimum linear estimation of stochastic signals in the presence of multiplicative noise," IEEE Trans. Aerospace Electron. Syst., 7, 462, 1971.

[11] Tugnait, J. K., "Stability of optimum linear estimators of stochastic signals in white multiplicative noise," IEEE Trans. Autom. Cont., 26, 757, 1981.

[12] DeKoning, W. L., "Optimal estimation of linear discrete-time systems with stochastic parameters," Automatica, 20, 113, 1984.

[13] Yaz, E., "Observer design for stochastic-parameter systems," Int. J. Cont., 46, 1213, 1987.

[14] Yaz, E., "Implications of a result on observer design for stochastic parameter systems," Int. J. Cont., 47, 1355, 1988.

[15] Nagpal, K. M., R. E. Helmick, and C. S. Sims, "Reduced order estimation: Part 1. Filtering," Int. J. Cont., 45, 1867, 1987.

ASYMPTOTIC PROPERTIES OF PARAMETER ESTIMATION FOR THE DMC MODELS

O.A. ASBJORNSEN* and K. YIN
Department of Chemical Engineering
University of Maryland, College Park, MD 20742
G. YIN†
Department of Mathematics
Wayne State University, Detroit, MI 48202

Abstract. The parameter identification problem for the Dynamic Matrix Control (DMC) model is considered in this work. A recursive algorithm for parameter estimation is developed. Some asymptotic properties of such an algorithm is obtained. It is shown that the algorithm is strongly consistent and a suitably scaled error sequence has normal limiting distribution. Consequently, the asymptotic normality is used to build up interval (confidence region) estimates.

Keywords. dynamic matrix control, consistency, asymptotic normality, confidence regions.

1. Introduction

The Dynamic Matrix Control (DMC) model introduced initially by Cutler *et al* [1, 2], and developed further by Garcia *et al* [3, 4] is a predictive controller which has been employed in petro-chemical industry with great success. This model can be used in a batch reaction as well as some continuous process, and is able to handle the process constraints successfully. As an optimal controller, this algorithm calculates moves on manipulated variables to create a change in the output vector that is a mirror image of the error vector about the setpoint.

In reality, the underlying system is usually subject to various disturbances, more often than not, only noise corrupted measurements are available. But to our knowledge the stochastic aspects of the DMC model have not been emphasized yet. Most developments have been centered around deterministic issues only. As far as the identification problem is concerned, the scope of the analysis and applications seem to be quite restricted. In fact, only independent zero mean Gaussian noise was considered, neither the consistency nor the asymptotic properties has been investigated. Identification for DMC remains to be the most time consuming step and requires expertise from the user in the implementation.

The objective of this work is to investigate asymptotic properties of the identification problem for the DMC model. First, by means of invoking pertinent results in least squares identification [5-9] for ARMA processes, strong consistency for the DMC model is obtained, which puts the analysis on a rigorous base. Next, an appropriately scaled error sequence is shown to have normal limiting distribution. In

---

[*] Research of this author has been supported in part by the Shell Development Company.

[†] Research of this author has been supported in part by the National Science Foundation under grant DMS-8814624, and in part by Wayne State University under the Wayne State University Research Award.

addition to point estimates, confidence interval estimates of the parameters are also obtained. The asymptotic normality is then used to construct confidence regions which are very useful in the future studies for various issues such as robustness control etc. The dimension of the confidence regions (e.g. the width of a confidence interval) is a measure of the overall quality of the estimation. The development of such confidence regions is essential for the determination of significance of the estimation as required for various hypotheses testing.

2. The DMC model

DMC is a discrete time, linear input-output model. It estimates the effect of past changes in manipulated variables and computes the control moves such that some performance criterion (for instance, a quadratic objective function) is minimized.

This model differs from the conventional optimal control formulation in that a 'moving horizon' is used in lieu of a fixed horizon. Thus, to some extent, the DMC can be thought of as an adaptive control procedure.

The dynamic matrix model is based on a convolution step-response formulation. For notational simplicity, we shall consider the single input and single output systems (SISO) only.

Let n denote the discrete time, q denote the order of the manipulated variable, $\{u_n\}$, $\{y_n\}$, $\{w_n\}$ denote the sequences of input, output and disturbance, respectively. Instead of using $\{u_n\}$ directly, we define the step-changes Δu_n as:

$$\Delta u_n = u_n - u_{n-1}.$$

The input-output model is given by:

$$y_n = y_0 + \sum_{k=1}^{q} a_k \Delta u_{n-k} + w_n, \tag{2.1}$$

where y_0 represents the initial condition.

From Eqn. (2.1), if the current time is $\overline{n}$, for any future time $\overline{n} + l$,

$$y_{\overline{n}+l} = y_0 + \sum_{k=1}^{q} a_k \Delta u_{\overline{n}+l-k} + w_{\overline{n}+l} \tag{2.2}$$

or

$$y_{\overline{n}+l} = y_0 + \sum_{k=l+1}^{q} a_k \Delta u_{\overline{n}+l-k} + \sum_{k=1}^{l} a_k \Delta u_{\overline{n}+l-k} + w_{\overline{n}+l} \tag{2.3}$$

the first two terms on the right-hand side of Eqn. (2.3) describe the effect of past control moves and the third term describe the effect of control moves after time $\overline{n}$ on the output.

In addition, the second term on the right-hand side of (2.1) can be written as $A^T \Delta u(n)$, where

$$A = \begin{pmatrix} a_1 \\ \vdots \\ a_q \end{pmatrix}, \quad \Delta u(n) = \begin{pmatrix} \Delta u_{n-1} \\ \vdots \\ \Delta u_{n-q} \end{pmatrix}.$$

In identification problems discussed hereof, A in (2.1) consists of the coefficients to be estimated. Actually, the"dynamic matrix" A in predictive control problem is one consists of all a_k's and has a more complicated structure (cf. [1]-[4]). All the a_k's must be available before the predictive control can be carried out.

Remark: It seems that the DMC model is very similar to the well-known ARMA model. The differences lie in, first, the increments of the inputs, instead of the inputs themselves are used, second, only the initial output, not the whole history of output, is included.

Reference [11] discussed the problem of random disturbances, i.i.d. white noise sequence was dealt with.

Since the coefficients in Eqn. (2.1) are generally not available, the first important task is to estimate them, i.e., identify each element of the dynamic matrix A. A common practice for doing this is by employing a least squares (or the like) procedure [11]. In most of the previous work for the identification of DMC models, only independent Gaussian random variables were considered. This is not quite satisfiable from both theoretical and practical points of view. To generalize the existing result to non-Gaussian dependent noise case, we allow the random disturbance to be of martingale difference type with 0 mean value.

For simplicity, and without loss of generality, we may assume $y_0 = 0$ henceforth. As a consequence, the underlying system can be described by

$$y_n = \sum_{k=1}^{q} a_k \Delta u_{n-k} + w_n \tag{2.4}$$

As a usual practice, denote

$$\theta^T = (a_1, a_2, \dots, a_q) \tag{2.5}$$

$$\phi_n^T = (\Delta u_n, \Delta u_{n-1}, \dots, \Delta u_{n-q+1}) \tag{2.6}$$

where z^T denotes the transpose of z. With these notations, Eqn. (2.4) now can be written as

$$y_n = \phi_{n-1}^T \theta + w_n. \tag{2.7}$$

To identify the unknown parameters, we shall use the following recursive least squares type of algorithm as in the case of ARMA model (cf. [5-9]).

Consider

$$\begin{aligned} \theta_{n+1} &= \theta_n + e_n P_n \phi_n (y_{n+1} - \phi_n^T \theta_n) \\ &= \theta_n + P_{n+1} \phi_n (y_{n+1} - \phi_n^T \theta_n) \end{aligned} \tag{2.8}$$

$$P_{n+1} = P_n - e_n P_n \phi_n \phi_n^T P_n \tag{2.9}$$

$$e_n = (1 + \phi_n^T P_n \phi_n)^{-1}. \tag{2.10}$$

In the above, P_n is assumed to be non-singular. For the case of ill-conditioned matrices, we refer the readers to [10].

Eqns. (2.8)-(2.10) are the recursive least squares estimates for the DMC models, In addition, a sequences of estimates for the variance σ^2 can be constructed as:

$$\sigma_n^2 = \frac{1}{n}\sum_{k=1}^{n}(y_k - \theta_n^1\phi_{k-1}^1 - \cdots - \theta_n^q\phi_{k-1}^q)^2$$

where θ_n^i, ϕ_k^i denote the ith components of θ_n and ϕ_k, respectively.

3. Asymptotic properties of the identification procedure

The objective of this section is to study the consistency and asymptotic properties of the aforementioned identification procedure. It is shown that the iterates converge with probability one (w.p.1) to the true parameter, and a scaled error sequence has a normal limiting distribution.

We assume the following conditions.

(A1) $\{w_n\}$ is a sequence of martingale difference with respect to an increasing sequence of σ-fields $\mathcal{F}_n$, such that $\sup_n E(|w_n|^\gamma \big| \mathcal{F}_{n-1}) < \infty$ w.p.1, for some $\gamma > 2$, and $E(w_n^2 \big| \mathcal{F}_{n-1}) \xrightarrow{n} \sigma^2$ w.p.1, for some constant $\sigma > 0$.

(A2) For each n, Δu_n is $\mathcal{F}_{n-1}$ measurable. In addition, for some $r > 0$, and w.p.1,

$$\lim_n \frac{\lambda_{\min}(\sum_{k=0}^n \phi_k\phi_k^T)}{\log n} = \infty,\ \lim_n \frac{\log \lambda_{\max}(\sum_{k=0}^n \phi_k\phi_k^T)}{n} = 0,\ |\Delta u_n| = O(n^r),$$

where $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ denote the minimal and maximum eigenvalues of A, respectively.

(A3) There exists a non-random matrix T_n, such that $T_nT_n^T$ is symmetric positive definite, and

$$T_n^{-1}(\sum_{k=0}^{n} \phi_k\phi_k^T)^{\frac{1}{2}} \xrightarrow{P} I \text{ and } \max_{1\le k\le n} |T_n^{-1}\phi_k| \xrightarrow{P} 0.$$

If $m(n)$ and $l(n)$ are sequences of positive integers tending to infinity, if

$$l(n)/m(n) \to d^2, \text{ for some } d > 0, \text{ then } T_{m(n)}^{-1}T_{l(n)} \xrightarrow{n} dI.$$

3.1 Consistency

Proposition 1. *Under assumptions (A1)-(A3),*

(1) The sequence of estimates θ_n defined by Eqns. (2.8)-(2.10) is strongly consistent, i.e., $\theta_n \xrightarrow{n} \theta$ w.p.1.

(2) $\sigma_n^2 \xrightarrow{n} \sigma^2$ w.p.1.

The essential idea of the proof is similar to that of [6, 8]. We omit the details.

Proposition 1 assures the convergence or consistency property of parameter estimates for DMC, it indicates that for large enough n, the estimate θ_n is sufficiently close to the true parameter θ. Furthermore, the sequence of estimations for the variance of $\{w_n\}$ is also strongly consistent in the aforementioned sense.

3.2 Asymptotic normality

The probability distribution of random variables θ_n is studied in this part. Since $\{w_n\}$ is not necessarily Gaussian, the best we can get is a central limit theorem type of result. We thus devote our attention to the limiting distribution of a scaled sequence $\{\theta_n - \theta\}$. The desired result is described by Theorem 1.

Theorem 1. *Under the assumptions of Proposition 1,* $\left(\sum_{k=0}^{n} \phi_k \phi_k^T\right)^{\frac{1}{2}} (\theta_{n+1} - \theta)$ *converges in distribution to a normal random variable with mean 0 and covariance* $\sigma^2 I$ *where* I *denotes the identity matrix.*

The proof of this theorem is in the light of the approach in [12]. Main ideas of the proof is provided in the appendix.

Our proof is different from the existing result of Ljung [13]. Firstly, a functional invariant theorem instead of the usual central limit theorem is obtained. Secondly, the weak convergence technique is used in the proof. The functional invariance theorem allows us to carry out operations such as random time changes etc. in a very convenient way.

4. Confidence regions

The result obtained in the previous section enables us to obtain expressions for the asymptotic covariance of the estimated quantities, which can in turn be used to compute the confidence intervals and to evaluate the reliability of estimates obtained.

4.1. Confidence ellipsoids

According to Proposition 1 and assumption (A3),

$$\sigma_n^{-2}(\theta_n - \theta)^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T\right) (\theta_{n+1} - \theta) \quad - \sigma^{-2}(\theta_n - \theta)^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T\right) (\theta_{n+1} - \theta) \xrightarrow{n} 0 \tag{4.1}$$

in probability. Moreover,

$$E_n = \left\{ \sigma_n^{-2}(\theta_{n+1} - \theta)^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T\right) (\theta_{n+1} - \theta) \leq c_\alpha \right\}$$

represents an ellipsoidal region with volume equal to

$$V(E_n) = \frac{\sigma^{\frac{q}{2}} n \pi^{\frac{q}{2}} c_\alpha^{\frac{q}{2}} [det(\sum_{k=0}^{n} \phi_k \phi_k^T)^{-1}]^{\frac{1}{2}}}{\Gamma(\frac{q}{2} + 1)}. \tag{4.2}$$

Since $\left(\sum_{k=0}^{n} \phi_k \phi_k^T\right)^{\frac{1}{2}} (\theta_{n+1} - \theta)$ is asymptotically normal,

$$\sigma^{-2}(\theta_{n+1} - \theta)^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T\right) (\theta_{n+1} - \theta) \xrightarrow{n} \chi_q^2$$

in distribution. By virtue of Proposition 1, Theorem 1, we arrive that

$$\begin{aligned} P(\theta \in E_n) &= \left\{ \sigma_n^{-2}(\theta_{n+1} - \theta)^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T\right) (\theta_{n+1} - \theta) \leq c_\alpha \right\} \\ &\xrightarrow{n} P(\chi_q^2 \leq c_\alpha) = 1 - \alpha. \end{aligned}$$

Therefore, E_n has limiting confidence level $1 - \alpha$. We summarize the above discussions in the following theorem.

Theorem 2. *Suppose that the assumptions (A1)-(A3) are fulfilled. Choose $\alpha > 0$, such that $P(\chi_q^2 \geq c_\alpha) = \alpha$. Then*

$$\lim_{n\to\infty} P(\theta \in E_n) = P\left\{ \sigma_n^{-2}(\theta_{n+1} - \theta)^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T \right) (\theta_{n+1} - \theta) \leq c_\alpha \right\} = 1 - \alpha.$$

Selecting a confidence level, from χ^2 distribution and Theorem 2, with prescribed estimation error, we can calculate the probability that the true parameter θ is within certain ellipsoid, so Theorem 2 can be used to establish a stopping rule for the identification procedure.

4.2. Confidence intervals for θ^i

In this subsection, we shall construct a $100\%(1-\alpha)$ percent confidence interval for each θ^i (component of θ), or equivalently the coefficients of the DMC. Such an approach will provide us with further insight along the line of the influence of each δ_i to the performance of the DMC model.

Let $\{e_1, \ldots, e_q\}$ be a standard basis for $\mathbb{R}^q$. In view of Theorem 1 and by virtue of Cramer-Wald's Theorem [14], for each $1 \leq i \leq q$,

$$e_i^T \left(\sum_{k=0}^{n} \phi_k \phi_k^T \right)^{\frac{1}{2}} (\theta_{n+1} - \theta) \xrightarrow{n} N(0, \sigma^2) \text{ in distribution.} \tag{4.3}$$

In order to obtain the desired result, one more assumption (A5) is needed.

(A4) For each n, Δu_n is $\mathcal{F}_{n-1}$ measurable. Moreover, there is a symmetric positive definite matrix R, such that

$$\frac{1}{n} \sum_{k=0}^{n} \phi_k \phi_k^T \xrightarrow{n} R \text{ w.p.1.} \tag{4.4}$$

Theorem 3. *If assumptions (A1) and (A4) are fulfilled, then*

(1) $\theta_n \xrightarrow{n} \theta$ *w.p.1;*

(2) $\sqrt{n}(\theta_{n+1} - \theta) \xrightarrow{n} N(0, \sigma^2 R^{-1})$ *in distribution.*

The proof of this theorem is sketched in the appendix.

With Theorem 3 at our disposal, we can choose $c_{\frac{\alpha}{2}}$ as the $100\%(1-\frac{\alpha}{2})$-th percentile, then for each $1 \leq i \leq q$.

$$\lim_{n\to\infty} P\left\{ \theta_{n+1}^i - c_{\frac{\alpha}{2}} \frac{\sigma R_{ii}^{-1}}{\sqrt{n}} \leq \theta^i \leq \theta_{n+1}^i + c_{\frac{\alpha}{2}} \frac{\sigma R_{ii}^{-1}}{\sqrt{n}} \right\} = 1 - \alpha$$

$$\lim_{n\to\infty} P\left\{ \theta_{n+1}^i - c_{\frac{\alpha}{2}} \frac{\sigma_n R_{ii}^{-1}}{\sqrt{n}} \leq \theta^i \leq \theta_{n+1}^i + c_{\frac{\alpha}{2}} \frac{\sigma_n R_{ii}^{-1}}{\sqrt{n}} \right\} = 1 - \alpha.$$

The above limits tell us how each component of θ_n is distributed in the limit. We are able to describe the probability of the deviation by standard normal distribution.

Moreover, confidence intervals for θ^i are obtained, such that the limiting confidence coefficient is equal to $1-\alpha$.

4.3. A stopping rule

In view of the recent development in stopping rules for least squares identification (cf. [12]), we propose the following for the DMC model.

Define

$$\tilde{V}_n = \frac{\sigma^{\frac{q}{2}}\pi^{\frac{q}{2}}c_{\alpha}^{\frac{q}{2}}[\det(T_n^{-1}(T_n')^{-1})]^{\frac{1}{2}}}{\Gamma(\frac{q}{2}+1)}.$$

Choose v_ϵ to be the smallest positive integer n, such that $\tilde{V}_n \leq \epsilon^q$; let τ_ϵ be the smallest positive integer n, such that $V(E_n) \leq \epsilon^q$. τ_ϵ is a stopping rule. Upon defining this random variable, one question arises immediately. Does this stopping rule have the desired properties? The answer to this question is given in the following theorem.

Theorem 4. *Under conditions of Theorem 2, we have*

$$\lim_{n\to\infty} \{\theta \in E_{\tau_\epsilon}, V(E_{\tau_\epsilon}) \leq \epsilon^q\} = 1-\alpha.$$

For a given $\epsilon > 0$ and a desired confidence level α. Whenever the iterate obtained from the recursive scheme is in an "θ-ellipsoid" with volume no larger than ϵ^q, the computation will be terminated with large probability.

5. Conclusion

A recursive identification algorithm for the DMC model is developed in this paper. Parameter estimation and identification for DMC model are considered within a probabilistic framework. Convergence properties are analyzed for parameter estimates θ_n of the DMC model. The asymptotic normality for the error sequence has been established. A confidence region approach for the underlying model was suggested. Finally, we would like to point out that:

(1) Extension to MIMO (multi-input multi-output) systems can be done in a similar manner. In stead of using Eqn. (2.1), we may consider the following input-output model: $y_n = \sum_{k=1}^{q} A_k \Delta u_{n-k} + w_n$, where $y_n, \Delta u_n, w_n$ are vectors and A_n are matrices with appropriate dimensions. The analysis is in the light of what presented here. The notations will be more complicated however.

(2) Further study can be directed toward various hypotheses testing problems. This in turn will enhance our understanding of the influence of various parameters to the performance and robustness of the controller.

Appendix

Due to the page limitation, we shall only indicate the salient feature of the proof with most of the details omitted.

Idea of Proof of Theorem 1: In lieu of working with the discrete expression directly, we shall take a continuous time interpolation. Let $[x]$ be the greatest integer which is less than or equal to x, and define

$$M_n(t) = T_n^{-1}\sum_{k=0}^{[nt]} \phi_k w_{k+1},\ t \in [0,\infty) \tag{a.1}$$

$$A_n(t) = T_n^{-1}\sum_{k=0}^{[nt]} \phi_k \phi_k^T E(w_{k+1}^2 \big| \mathcal{F}_k)(T_n^{-1})^T,\ t \in [0,\infty). \tag{a.2}$$

We first show that if the conditions of Theorem 2 are satisfied, then $M_n(\cdot) \Rightarrow M(\cdot)$, a process with independent Gaussian increments and continuous sample paths.

Observe that

$$M_n(\cdot) \in D^q[0,\infty),\ A_n(\cdot) \in D^{q\times q}[0,\infty),$$

and $M_n(0) = A_n(0) = 0$, where these D spaces are spaces of functions which are right continuous and have left limits endowed with the Skorohod topology ([14]-[16]) for various definitions and basic theorems of weak convergence). By virtue of assumptions (A1) and (A3), $A_n(t) \xrightarrow{n} \sigma^2 tI$ in probability.

To proceed, we consider the truncated processes defined as follows. Let ν be a positive real number, and let $\gamma_\nu^n = \inf\{t; |M_n(t)|^2 > \nu, \text{ or } |M_n(t^-)|^2 > \nu\}$, $M_n^\nu(t) = M_n(t \wedge \gamma_\nu^n)$ and $A_n^\nu(t) = A_n(t \wedge \gamma_\nu^n)$, where $(a \wedge b) = \min(a,b)$. Let M^ν be any limit point of M_n^ν and

$$\gamma_\nu = \inf\{t; |M(t)|^2 > \nu, \text{ or } |M(t^-)|^2 > \nu\}.$$

Then, $A_n^\nu(t) \xrightarrow{P} \sigma^2(t \wedge \gamma_\nu)I$. Moreover, both $M^\nu(\cdot)$ and $A^\nu(\cdot)$ are uniformly integrable.

Let $M_n^{\nu,i}(\cdot)$ denote the ith component of $M_n^\nu(\cdot)$. Then, the quadratic variation is given by

$$\tilde{A}_n^{ij}(t) = [M_n^{\nu,i}(t), M_n^{\nu,j}(t)] = \sum_{k=0}^{[n(t\wedge\gamma_\nu^n)]} (T_n^{-1}\phi_k)^i (T_n^{-1}\phi_k)^j w_{k+1}^2.$$

Moreover, $A_n^\nu(\cdot)$ and $\tilde{A}_n(t)$ have the same limit in probability; $M_n^{\nu,i}(t)M_n^{\nu,j}(t) - \tilde{A}_n^{ij}(t)$ are martingales.

Next, we verify the following equation holds. For each $T > 0$,

$$\lim_n E\left(\sup_{t\in[0,T]} |M_n^\nu(t) - M_n^\nu(t^-)|\right) = 0. \tag{a.3}$$

Now, all conditions of Theorem 7.1.4 in Ethier and Kurtz [15] are verified. By applying that theorem, $M_n^\nu(\cdot)$ converges weakly to $M^\nu(\cdot)$, a process with independent Gaussian increments and continuous paths. Next, by virtue of Theorem

7.1.1 in [15], there exists a unique (in the sense of distribution) process $M(\cdot)$ with continuous paths, such that $M^i(t)M^j(t) - A^{ij}(t)$ are martingales, and $M(\cdot)$ has independent Gaussian increments.

Finally, letting $\nu \to \infty$, similar to the approach in Kushner [16], we can show that $M_n(\cdot) \Rightarrow M(\cdot)$.

Setting $t = 1$ in (a.1), $M_n(1) \xrightarrow{D} N(0, \sigma^2 I)$. Furthermore, it can be proven

$$\left(\sum_{k=0}^{n} \phi_k \phi_k^T \right)^{\frac{1}{2}} (\theta_{n+1} - \theta) \sim N(0, \sigma^2 I).$$

Proof of Theorem 3. The strong consistency can be proved by modifying the argument of Ljung ([13, 17]). As for the asymptotic normality, we shall modify our proof for Theorem 1 to obtain the desired result. Notice that

$$\sqrt{n}(\theta_{n+1} - \theta) = \left(\frac{1}{n} \sum_{k=0}^{n} \phi_k \phi_k^T \right)^{-1} \frac{1}{\sqrt{n}} \sum_{k=0}^{n} \phi_k w_{k+1}.$$

Due to Eqn. (4.4) and Cramer-Wald's Theorem, we need only show that

$$\frac{1}{\sqrt{n}} \sum_{k=0}^{n} \phi_k w_{k+1} \xrightarrow{n} N(0, \sigma^2 R) \text{ in distribution .}$$

In lieu of definitions (a.1) (a.2), define

$$M_n(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{[nt]} \phi_k w_{k+1}, \; A_n(t) = \frac{1}{n} \sum_{k=0}^{[nt]} \phi_k \phi_k^T E(w_{k+1}^2 | \mathcal{F}_k), \; t \in [0, \infty).$$

Proceed as in the proof of Theorem 1, the desired result follows.

Proof of Theorem 4. In fact, we need only show the asymptotic normality holds with the deterministic time n replaced by τ_ϵ, with $\epsilon \to 0$. Owing to the definition of $M_n(\cdot)$, we need only show that $M_{\tau_\epsilon}(\cdot)$ converges weakly to $M(\cdot)$ with independent Gaussian increments and continuous paths.

To this end, define

$$\Phi_{v_\epsilon}(t) = \begin{cases} \frac{\tau_\epsilon}{v_\epsilon}, & \text{if } \frac{\tau_\epsilon}{v_\epsilon} \le 1; \\ t, & \text{otherwise .} \end{cases}$$

By virtue the definition of v_ϵ and τ_ϵ, $\lim_{\epsilon \to 0} P\{\tau_\epsilon / v_\epsilon > 1\} = 0$, and $v_\epsilon \to \infty$, $\tau_\epsilon \to \infty$, as $\epsilon \to 0$. By Lemma a.1, $M_{v_\epsilon}(t)$ converges to $M(t)$ weakly as $\epsilon \to 0$, and hence $M_{v_\epsilon}(\Phi_{v_\epsilon}(t))$ converges weakly to $M(t)$. By virtue of an argument as in [12], it can be shown $|M_{\tau_\epsilon}(t) - M_{v_\epsilon}(\Phi_{v_\epsilon}(t))| \to 0$ in probability. This yields the result.

References

[1] C. R. Cutler and B. L. Ramaker, Dynamic matrix control - a computer algorithm, *AICHE 86th National Meeting*, 1979, Paper 51b.

[2] C.R. Cutler, Dynamic matrix control, an optimal multivariable control algorithm with constraints, *Ph.D Dissertation*, University of Houston, 1983.

[3] C.E. Garcia and A.M. Morshedi, Quadratic programming solution of dynamic matrix control (QDMC), *Chem. Eng. Comm.* vol. 46(1986), 73-87.

[4] D. M. Prett and C. E. Garcia, *Fundamental Process Control*, Butterworths, Boston, 1988.

[5] G.C. Goodwin and R.L. Payne, *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, New York, 1977.

[6] T.Z. Lai and C.Z. Wei, Least square estimations in stochastic regression models with applications to identification and control of dynamic systems, *Ann. of Statist.*, vol. 10(1982), 154-166.

[7] H.F. Chen and L. Guo, Convergence rate of least squares identification and adaptive control for stochastic systems, *Internat. J. Control*, 44(1986), 1459-1476.

[8] H. F. Chen, *Recursive Estimation and Control for Stochastic Systems*, John Wiley & Sons, New York, 1985.

[9] P.R. Kumar and P.P. Varaiya, *Stochastic Systems: Estimation, Identification and Adaptive Control*, Prentice-Hall, Englewood Cliffts, NJ, 1986.

[10] O.A. Asbjornsen, Stability and robustness of the DMC algorithm as compared with the traditional ARMA model, *Proc. ACC*, 1988, 278-283.

[11] O.A. Asbjornsen, On linear parameter estimation by sequential partial regression, *AIChE Annual Meeting*, 1984, paper 82g.

[12] G. Yin, A stopping rule for least squares identification, *IEEE Trans. Automat. Control*, AC-33 (1989), 659-662.

[13] L. Ljung, *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffts, NJ, 1987.

[14] P. Billingsley, *Convergence of Probability Measures*, J. Wiley & Sons, New York, 1968.

[15] S. Ethier and T.G. Kurtz, *Markov Processes*, J. Wiley & Sons, New York, 1986.

[16] H.J. Kushner, *Approximation and Weak Convergence Methods for Random Processes with Applications to Stochastic Systems Theory*, MIT Press, Cambridge, MA, 1984.

[17] L. Ljung, Consistency of the least squares identification method, *IEEE Trans. Automat. Control*, AC-24(1976), 779-781.

[18] O.A. Asbjornsen, K. Yin and G. Yin, On confidence region approach to DMC models, preprint.

Adaptive Control of A Class of Nonminimun Phase Plant

Part I: Proof of Stability

Shi Bai

Department of Electrical Engineering
Michigan State University
East Lansing, MI 48824

ABSTRACT

One of the basic assumptions in model reference adaptive control is that the transfer function of the modeled plant is minimum phase. In this paper we relax this assumption by developing a model reference adaptive controller which has the precompensator and feedback controller as usual, and it also has a feedforward controller. The stability and convergence proof is presented.

1. Introduction

In the field of adaptive control, a problem which has attracted considerable recent interest is the analysis of robustness properties. Considerable progress has been made on the issues of external perturbations, unmodeled dynamics, time-varying parameters, and certain forms of plant nonlinearities [1-9].

In the design and analysis of model reference adaptive control scheme [10-13], the assumptions on the transfer function of the modeled part of plant which lead to complete the stability and convergence proof of several classes of model reference adaptive control schemes in 1970's are:

A1. The number of poles is known.

A2. The number of zeros is known.

A3. The sign of the high frequency gain is known.

A4. No disturbances and unmodeled dynamics.

A5. The transfer function is minimum phase. i.e. the zeros lie in the open left half plane (C_0^-).

Perhaps the most unrealistic among the (ideal) assumptions are the ones that there are no disturbances and unmodeled dynamics and that the transfer function of the modeled part of plant is minimum phase. The necessity of these restrictive assumptions needed to complete the stability and convergence proofs has triggered new research into the robustness of model reference adaptive control schemes. Analysis and simulation show that when an adaptive control system with the original (ideal) adaptation law is perturbed by some disturbances, and/or there are unmodeled dynamics affecting the system, several types of instabilities may arise in the system [4].

For relaxing the assumption A4 several modified adaptation schemes have been proposed. However, the assumption A5 is a essential assumption in the proof of the existence of a such control and the proof of stability and convergence. The assumption A5 still remain in the current adaptive control literature and research. How to relax the (ideal) assumption A5 is still an open question.

In this paper we relax the (ideal) assumption A5 on the transfer function of the modeled part of plant. We restrict our attention here on the case of no disturbances and unmodeled dynamics. the robustness issue with respect to disturbances and unmodeled dynamics is more involved and is reported in [14]. We construct a new adaptive controller structure. And we give the stability and convergence proof.

The paper is organized as follows: In section 2, we consider a general linear time-invariant SISO nonminimum phase plant and construct the controller structure. In section 3, we give the stability and convergence proof. In section 4, we conclude some final remarks.

2. General Nonminimum Phase Plant

We consider a SISO nth-order linear time-invariant plant. The transfer function of the plant is

$$W_p(s) = K_p \frac{Z_p(s)}{P_p(s)} \tag{1}$$

where $Z_p(s)$ and $P_p(s)$ are monic coprime polynomials of degrees m and n $(n > m)$ respectively. The degrees m and n and the sign of K_p are known, and $Z_p(s)$ is known and nonminimum phase.

A minimum realization of the plant is given as follows

$$\dot{x}_p = A_p x_p + b_p u_p, \qquad y_p = h_p^T x_p, \tag{2}$$

where $x_p \in \mathbf{R}^n$ is the state of the plant, $u_p \in \mathbf{R}^1$ is the input of the plant, $y_p \in \mathbf{R}^1$ is the output of the plant.

The transfer function of the reference model is

$$W_m(s) = K_m \frac{Z_m(s)}{P_m(s)}, \tag{3}$$

where $Z_m(s)$ and $P_m(s)$ are monic Hurwitz polynomials of degrees m and n respectively. For ease of presentation and without loss of generality, we assume that $n - m = 1$ and $K_p = K_m = 1$.

we rewrite the equation (3) as follows:

$$W_m(s) = \frac{Z_p(s)}{P_m(s)} + \frac{Z_m(s) - Z_p(s)}{P_m(s)} \tag{4}$$

A minimal realization of $\dfrac{Z_p(s)}{P_m(s)}$ is

$$\dot{x}_m = A_m x_m + b_m r, \qquad y_m = h_m^T x_m, \tag{5}$$

A minimal realization of $\dfrac{Z_m(s) - Z_p(s)}{P_m(s)}$ is

$$\dot{\eta}_m = A_m \eta_m + b_\eta r, \qquad y_{\eta_m} = h_\eta^T \eta_m, \tag{6}$$

where $x_m, \eta_m \in \mathbf{R}^n$, $r \in \mathbf{R}^1$ is the reference input, $y_m, y_{\eta_m} \in \mathbf{R}^1$.

We use the input u_p and output y_p to generate a $(2n - 1)$– dimensional auxiliary vector as

$$\dot{\omega}_1 = F\omega_1 + q u_p, \qquad \dot{\omega}_2 = F\omega_2 + q y_p, \tag{7}$$

where F is a stable matrix.
Define the vectors

$$\omega^T = [r \ , \omega_1^T \ , y_p \ , \omega_2^T]$$

and

$$\theta^T = [1 \ , c^T \ , d_0 \ , d^T].$$

Then, the controller input is given as follows

$$u_p = \theta^T \omega. \tag{8}$$

We construct a feedforward compensator

$$\begin{aligned} \dot{\eta}_p &= A_m \eta_p + b_\eta r, \\ y_{\eta_p} &= h_\eta^T \eta_p, \end{aligned} \tag{9}$$

And, we also define the output error as

$$e_1 = y_p - y_m, \quad e_\eta = y_{\eta_p} - y_{\eta_m}. \tag{10}$$

Now we have

$$\dot{X}_c = \begin{bmatrix} \dot{x}_p \\ \dot{\omega}_1 \\ \dot{\omega}_2 \end{bmatrix} = \begin{bmatrix} A_p & 0 & 0 \\ 0 & F & 0 \\ q h_p{}^T & 0 & F \end{bmatrix} \begin{bmatrix} x_p \\ \omega_1 \\ \omega_2 \end{bmatrix} + \begin{bmatrix} b_p \\ q \\ 0 \end{bmatrix} u_p. \tag{11}$$

The adaptive controller structure should have enough degrees of freedom so that with the time varying parameters c^T, d_0 and d^T set equal to constants, and by suitable choice of F and q, the overall plant and controller transfer function matches that of the specified model. This will be verified in the following Theorem under the condition that the model has the same relative degree as the plant. First, notice that F and q enter our calculations only through the transfer function $(sI - F)^{-1} q$ so that we may assume without loss of generality that F and q are in controllable form so that

$$(sI - F)^{-1} q = \frac{1}{\hat{F}(s)} \begin{bmatrix} 1 \\ s \\ \cdot \\ \cdot \\ s^{n-2} \end{bmatrix} \tag{12}$$

with $\hat{F}(s) = \det(sI - F) \in \mathbf{R}(s)$. To keep the controller stable we will insist on $\hat{F}(s)$ being Hurwitz: from (12), we note that the choice of F and q is now equivalent to the choice of a stable $\hat{F}(s) \in \mathbf{R}(s)$.

Theorem: Given a model $W_m(s)$ of the form (3) and plant of form (1), there exists a choice of $d_0^* \in \mathbf{R}$, $c^*, d^* \in \mathbf{R}^{n-1}$, $\hat{F}(s) \in \mathbf{R}(s)$ and Hurwitz so that the closed loop transfer function of the plant and controller equals $W_m(s)$.

Proof: when the adaptive gains are constant with values given by c^*, d_0^* and d^* the transfer function of the plant and controller is given by

$$\frac{Z_p(s)\hat{F}(s)}{[\hat{F}(s) - \hat{C}^*(s)]P_p(s) - \hat{D}^*(s)Z_p(s)} + \frac{Z_m(s) - Z_p(s)}{P_m(s)} \tag{13}$$

with $\hat{C}^*(s) = c_1^* + c_2^* s + \cdots + c_{n-1}^* s^{n-2}$ has degree $n-2$
and $\hat{D}^*(s) = d_0\hat{F}(s) + d_1^* + d_2^* s + \cdots + d_{n-1}^* s^{n-2}$ has degree $n-1$.

To show that the expression in (13) can be made equal to $W_m(s)$ for suitable choice of $\hat{C}^*(s)$ and $\hat{D}^*(s)$ we need the following lemma.

Lemma: Given two coprime monic polynomials $Z_p(s)$ and $P_p(s)$ of degree m and n, and an arbitrary monic polynomial $U(s)$ of degree $2n-1$, there exist unique polynomial $T(s)$ (monic) and $R(s)$ of degree $n-1$ respectively so that

$$T(s)P_p(s) + R(s)Z_p(s) \equiv U(s). \tag{14}$$

Proof:

Existence:

Since $P_p(s)$ and $Z_p(s)$ are coprime, it follows from the Euclidean algorithm that there exist polynomials $A(s)$ of degree $n-1$ and $B(s)$ of degree $m-1$ such that

$$A(s)Z_p(s) + B(s)P_p(s) \equiv 1. \tag{15}$$

Multiplying both sides of (15) by $U(s)$ we have

$$U(s)A(s)Z_p(s) + U(s)B(s)P_p(s) \equiv U(s). \tag{16}$$

Dividing $U(s)A(s)$ by $P_p(s)$ we get

$$\frac{U(s)A(s)}{P_p(s)} = Q(s) + \frac{R(s)}{P_p(s)} \tag{17}$$

where $Q(s)$ is the quotient and $R(s)$ is the remainder of degree $\leq n-1$. Now we rewrite (16) as

$$[U(s)A(s) - Q(s)P_p(s)]Z_p(s) + [U(s)B(s) + Q(s)Z_p(s)]P_p(s) \equiv U(s). \tag{18}$$

Using (17) we see that (18) is of the required form (14) since $U(s)A(s) - Q(s)Z_p(s) = R(s)$ has degree $\leq n-1$; thereby forcing $T(s) = U(s)B(s) - Q(s)Z_p(s)$ to have degree $\leq n-1$. Further, $U(s)$ monic implies that $T(s)$ is monic.

Uniqueness:

If there exist other polynomials $T_1(s)$ (monic) and $R_1(s)$ of degree $(n-1)$ so as to satisfy (14) we would have

$$[T(s) - T_1(s)]P_p(s) + [R(s) - R_1(s)]Z_p(s) \equiv 0$$

so that

$$\frac{Z_p(s)}{P_p(s)} = -\frac{[T(s) - T_1(s)]}{[R(s) - R_1(s)]}$$

with $R(s) - R_1(s)$ of degree $n-1$, thereby contradicting the coprimeness of $Z_p(s)$ and $P_p(s)$.

To apply the lemma choose

$$U(s) = P_m(s)\hat{F}(s)$$

Verify that $U(s)$ is a monic polynomial of degree $2n-1$.

Now, by the lemma, there exist suitable choice of $\hat{C}^*(s)$, $\hat{D}^*(s)$ and $\hat{F}(s)$ such that the expression of (13) can be made equal to

$$[\hat{F}(s) - \hat{C}^*(s)]P_p(s) - \hat{D}^*(s)Z_p(s) = P_m(s)\hat{F}(s) \tag{19}$$

And

$$\frac{Z_p(s)\hat{F}(s)}{[\hat{F}(s) - \hat{C}^*(s)]P_p(s) - \hat{D}^*(s)Z_p(s)} + \frac{Z_m(s) - Z_p(s)}{P_m(s)} = \frac{Z_m(s)}{P_m(s)} \tag{20}$$

where $P_p(s)$ and $Z_p(s)$ are monic, Hurwitz polynomials of degree n and m respectively, and $\hat{F}(s)$ can be any monic Hurwitz polynomial.

3. Stability of The Adaptive System

In this section we apply the results of section 2 to analyze the stability of the adaptive system in section 2. It has been shown that there exists

$$\theta^* = [1, c^{*T}, d_0^*, d^{*T}]^T$$

so that the transfer function of the plant and controller can be made equal to that of the reference model when $\theta = \theta^*$.

Now define the parameter error as follows

$$\phi = \theta - \theta^* \tag{21}$$

Thus the control input can be expressed as

$$u_p = \phi^T \omega + r + c^{*T} \omega_1 + d_0^* h_p^T x_p + d^{*T} \omega_2 \tag{22}$$

Rewrite equation (11) as

$$\dot{X}_c = \begin{bmatrix} A_p + d_0^* b_p h_p^T & b_p c^{*T} & b_p d^{*T} \\ d_0^* q h_p^T & F + q c^{*T} & q d^{*T} \\ q h_p^T & 0 & F \end{bmatrix} X_c + \begin{bmatrix} b_p \\ q \\ 0 \end{bmatrix} (r + \phi^T \omega). \tag{23}$$

So,

$$\dot{X}_c = A_c X_c + b_c(\phi^T \omega + r) \tag{24}$$

$$y = h_c^T X_c \tag{25}$$

and the augmented reference model from (5) is given by

$$\dot{X}_{mc} = A_c X_{mc} + b_c r \tag{26}$$

$$y_m = h_c^T X_{mc}. \tag{25}$$

Define

$$e = X_c - X_{mc}. \tag{26}$$

Then,

$$\dot{e} = A_c e + b_c \phi^T \omega, \tag{27}$$

$$e_1 = h_c e. \tag{28}$$

Now, we define the adaptation law as

$$\dot{\phi} = \dot{\theta} = -e_1 \omega \tag{29}$$

For the purpose of a stability proof one needs to assume that the transfer function of the model $W_m(s)$ is Strictly Positive Real (SPR). This entails no loss of generality. Hence, by the Positive Real Lemma, there exist positive definite matrices P_c, Q_c such that

$$A_c^T P_c + P_c A_c = -Q_c,$$

$$P_c b_c = h_c. \tag{30}$$

Now the system under consideration is given by

$$\dot{e} = A_c e + b_c \phi^T \omega, \tag{31}$$

$$\dot{\phi} = = -e_1 \omega \tag{32}$$

Choose the positive definite (Lyapunov) function

$$V(e, \phi) = e^T P e + \phi^T \phi \tag{33}$$

Then,

$$\dot{V}(e, \phi) = -e^T Q e \le 0. \tag{34}$$

Since $\dot{V} \le 0$; we have that $e(t)$, $\phi(t)$ are bounded. And, it is easy to show that the error $e \to 0$ as $t \to \infty$. We see also from (32)

$$\lim_{t \to \infty} \phi = 0$$

and from (33) and (34) that

$$\lim_{t \to \infty} V(e, \phi) = \lim_{t \to \infty} \phi^T \phi = V^*$$

exist. Though $||\phi(t)||$ converges to $(V^*)^{\frac{1}{2}}$, we can say nothing about the convergence of $\phi(t)$ as $t \to \infty$. A sufficient rich reference input signal is needed to obtain the parameter convergence and to achieve the uniformly asymptotically stability.

4. Conclusion

We have presented some initial results on adaptive control of nonminimum phase plant. The main contribution of this paper is a new adaptive controller structure and the stability proof of adaptive control for a class of nonminimum phase plant. Since the transfer function of the modeled part of plant is often obtained in form of nonminimum phase, a stable adaptive control of nonminimum phase plant is a very important issue in the development of adaptive control theory as well as in the practical application. The stability proof of such a class of nonminimum phase plant is presented based on the assumptions A1-A4. The robustness issue of this adaptive control system will be discussed in [14]. And, the simulation results will be provided later.

5. References

[1] B. B. Peterson and K. S. Narendra, "Bounded error adaptive control," *IEEE Trans. Automat. Contr.*, Vol. AC-27, No. 6, pp. 1161-1168, 1982.

[2] B. Riedle, B. Cyr and P. V. Kokotovic, "Disturbance instability in an adaptive system," *IEEE Trans. Automat. Control* 29 (1984) pp. 822-824.

[3] P. A. Ioannou and P. V. Kokotovic, "Robust redesign of adaptive control," *IEEE Trans. Automat. Contr.*, Vol. AC-29, No. 3, 1984.

[4] P. I. Ioannou and P. V. Kokotovic, "Instability analysis and robustness of adaptive control," *Automatica*

[5] C. E. Rohrs, L. Valavani, M. Athans and G. Stein, "Robustness of continuous-time adaptive control algorithms in the presence of unmodeled dynamics," *IEEE Trans. Automat. Control* 30 (1985) pp. 881-889.

[6] R. L. Kosut, B. D. O. Anderson and I. M. Y. Mareels, "Stability Theory for Adaptive Systems: Method of Averaging and Persistency of Excitation," *IEEE Trans. Automat. Control* 32 (1987) pp. 27-34.

[7] K. S. Narendra and A. M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 134-145, 1987.

[8] Shi Bai and Fathi M. A. Salam, "A robust adaptive controller for linear time-invariant plants with external disturbances," *1989 American Control Conference* (ACC).

[9] S. S. Sastry and A. Isidori, "Adaptive Control of Linearizable Systems," Technical Report, Memo. No. UCB/ERL M87/53.

[10] K. S. Narendra, Y. H. Lin and L. S. Valavani, "Stable adaptive controller design, Part II: Proof of stability," *IEEE Trans. Automat. Control* 25 (1980) pp. 440-448.

[11] S. Sastry, "Model reference adaptive control - stability, parameter convergence, and robustness," *IMA J. Math. Control Inform.* 1 (1984) pp. 27-66.

[12] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems* Prentice Hall, 1989

[13] Shanker Sastry and Marc Bodson, *Adaptive Control* Prentice Hall, 1989

[14] Shi Bai, "Adaptive Control of Nonminimum Phase Plant Part II: Robustness Issue," , under preparation.

RECURSIVE ESTIMATION USING PIPELINE STRUCTURE

G. YIN*

Department of Mathematics
Wayne State University, Detroit, MI 48202

Abstract. A recursive estimation algorithm of stochastic approximation type with pipeline structure is proposed and analyzed. In lieu of using a single processor alone as in the classical setting, a collection of parallel processors are employed to iterate on the same system. The processors are arranged as on a production line. Certain limit theorems are obtained by means of weak convergence methods and martingale averaging technique. It is shown that suitably interpolated sequences converge either to an ordinary differential equation or a stochastic differential equation.

Keywords. recursive estimation, pipeline, parallel processing, weak convergence.

1. Introduction

In this work, we consider stochastic approximation type of recursive estimation algorithms. The emphasis is on the utilization of pipelining structure. Our goal is to design real time implementable procedures.

Recently, a considerable amount of attention has been drawn to the study of parallel and distributed recursive estimation problem of the stochastic approximation type. Progress has been made for investigating asymptotic properties of such procedures (cf. [1-9] and the references therein).

A decentralized stochastic approximation algorithm was first proposed in [1-2]. Such an algorithm uses a collection of parallel processors to estimate the same system; the participating processors communicate with each other through convexification. This algorithm was developed further in [3], rates of convergence, state dependent noise etc. were treated. In [4], another type of parallel algorithms was suggested, which is quite natural and arises when parallel processing methods are used for the Monte Carlo optimization of systems. In that algorithm, different processors control different components of the state; they compute and communicate in an asynchronous way and at 'renewal times'. This class of procedures was analyzed thoroughly in [4]. Later on, a specialized version of the algorithm discussed in [4] was studied in [5-6]. Questions, such as strong consistency and robustness were answered by employing the randomly varying truncation methods.

There are certain advantages of the aforementioned algorithms. By using such algorithms, we are able to exploit the opportunities provided by parallel processing, to utilize state space decomposition, and to take advantage of asynchronous communications. In a wide range of applications, one may encounter the situation that several physically separated processors (estimators, trackers, identifiers etc.)

---

[*] This research has been supported in part by the National Science Foundation under grant DMS-8814624, and in part by Wayne State University under the Wayne State University Research Award.

acting on or following the same system. Therefore, recursive estimations by means of parallel processing methods comes into play naturally. It is conceivable that due to the rapid advances in science and technology, parallel processing will play more and more important role in the future.

In the present work, we concern ourselves with the problem of design real time implementable algorithms. Such an idea was discussed in a recent paper [7]. We shall extend the result to a more general non-additive noise case, and to obtain some limit theorems by means of weak convergence methods.

The rest of the paper is organized as follows. In the next section, the problem formulation is given, and the algorithm with pipeline structure is proposed. Taking appropriate interpolation, we obtain a limit theorem in section 3. In preparing for further investigation of the asymptotic properties, we derive some order of magnitude estimates in section 4. A sequence of suitably normalized errors is then examined in section 5, and a limiting SDE (stochastic differential equation) is obtained. Finally, we conclude this paper by making some additional remarks in section 6.

2. The algorithm

Let $x, \xi \in \mathbb{R}^r$, $b(\cdot,\cdot) : \mathbb{R}^r \times \mathbb{R}^r \mapsto \mathbb{R}^r$. The classical recursive estimation algorithm is of the form

$$X^{\varepsilon}_{n+1} = X^{\varepsilon}_n + \varepsilon b(X^{\varepsilon}_n, \xi^{\varepsilon}_n), \tag{2.1}$$

where $\varepsilon > 0$ is known as the step-size. Here, we take the step-size to be a constant. Other forms of step-size can be considered. For example, in lieu of ε, one may take a sequence of decreasing positive numbers ε_n, with $\varepsilon_n \to 0$ and $\sum_n \varepsilon_n = \infty$. In applications, for example in tracking on-line estimation, one may wish to use constant gain since such algorithms seem to be rather robust with respect to large burst of noise.

A word about the notations is in order. In what follows, we often omit the ε dependence for notational simplicity, and retain it whenever necessary. The symbol "$'$" denotes the transpose of a matrix, and "$\Rightarrow$" stands for weak convergence. K will be used as a generic positive constant.

The time indicator n in (2.1) represents the iteration number rather than the real time. More often than not, the real time elapsed in one iteration is not equal to one unit. In fact, two phase operations are involved in (2.1). In phase 1, a processor takes an noisy observation, and in phase 2, a new value is obtained by adding its current value to the newly obtained observation. The second phase is usually not as time consuming as the first one. This real time implementation issue is not, however, reflected in algorithm (2.1). To take into account of this matter, we propose a new algorithm.

Let us look at the following example first. Suppose that the required time for taking a measurement is 3 units of real time, and the time needed for the addition of X_n and $b(X_n, \xi_n)$ is 1 unit. Instead of using a single processor alone to implement the computation, three parallel processors are employed in a pipeline fashion. All processors perform the same kind of computation, but with different starting time. Let n denote discrete real time. At a given instance n, suppose that processor 1 has completed its measurement started three time units ago, i.e., $b(X_{n-3}, \xi_{n-3})$ has been obtained. Suppose that X_n is available which was communicated by processor 3. Then, phase 2 is executed for processor 1, i.e.,

$$X_{n+1} = X_n + \varepsilon b(X_{n-3}, \xi_{n-3}).$$

At the next time instance $n+1$, processor 1 sends its content X_{n+1} to processor 2. For simplicity, we assume that the passage of information is instantaneous, i.e., there is no time delay. By now, processor 2 has completed its measurement started at $n-2$. Then,

$$X_{n+2} = X_{n+1} + \varepsilon b(X_{n-2}, \xi_{n-2}),$$

and the result X_{n+2} is passed to processor 3, etc.

In view of the above illustration, the overall system consists of a number of parallel processors connected through asynchronous communications. A sequence of estimates is generated by using a number of processors cyclically and periodically. The length of a cycle is equal to the number of processors participating in the computation, which is also equal to the time required for a single processor to complete its phase 1 operation.

In fact, all processors perform the same kind of operation with different starting time. Hence, it is enough to describe the algorithm by using a single formula. Suppose that it takes d unit of time for a single processor to complete one iteration. We use d processors in a concurrent way. Consider the following algorithm:

$$X^\varepsilon_{n+1} = X^\varepsilon_n + \varepsilon b(X^\varepsilon_{n-d}, \xi^\varepsilon_{n-d}). \tag{2.2}$$

In (2.2), the observation appeared to be in a form with delays. The delay however, is more form than substance. All that says is that the observation was started d units ago. We refer to such observations as "delayed" observations.

To demonstrate the algorithm works, we need to investigate the problem of convergence, rates of convergence etc. Due to the constant step-size, the pertinent limit theorem is in the sense of weak convergence (cf. [11-13]). We shall use mainly the weak convergence and martingale averaging technique developed in [11, 14, 15] to carry out the analysis.

3. Some limit results

We make the following assumptions.

(A1) $b(\cdot,\cdot) : \mathbb{R}^r \times \mathbb{R}^r \mapsto \mathbb{R}^r$ is continuous. $b_x(\cdot,\xi), b_{xx}(\cdot,\xi)$ are continuous uniformly in ξ. For each $T < \infty$, and each N, the set

$$\left\{ \sup_{|x|\le N} |b(x,\xi_j)|; \varepsilon > 0, j \ge 0, j\varepsilon \le T \right\}$$

is uniformly integrable, where $b_x(\cdot,\xi)$, $b_{xx}(\cdot,\xi)$ denote the first and the second derivatives of $b(\cdot,\xi)$, respectively.

(A2) There exists a function $\bar{b}(x)$, which is twice continuously differentiable, such that for each x, each m, the following holds

$$\frac{1}{n}\sum_{j=m}^{n+m} E_m b(x,\xi_j) \xrightarrow{n} \bar{b}(x) \text{ in probability}, \tag{3.1}$$

where E_m denotes the conditioning on the past up to m.

To study the limit behavior, we take a piecewise constant interpolation as follows. Let $x^\varepsilon(t) = X_n^\varepsilon$ for $t \in [n\varepsilon, n\varepsilon + \varepsilon)$. We shall show $x^\varepsilon(\cdot)$ so defined converges weakly to a process $x(\cdot)$ which satisfies an ODE (ordinary differential equation).

Remark: As far as the convergence alone is concerned, weaker conditions are possible. For example, the smoothness assumption (A1) can be replaced a "weak continuity". Similarly, (A2) can take weaker form as well. We are using the current setting for notational simplicity. Our main interest in this paper is to describe the new algorithm not to find the weakest condition. The proofs of the following theorems are similar to that of [11, 14, 15]. Cares, however, must be taken for the "delays".

Theorem 3.1. *Suppose (A1) (A2) are satisfied, and equation (3.2) has a unique solution for each initial condition* x_0. *Let* $X_j^\varepsilon \Rightarrow x_0$, $j = 0, \dots, r-1$. *Then,* $\{x^\varepsilon(\cdot)\}$ *is tight, and any convergent subsequence satisfies*

$$\dot{x} = \bar{b}(x). \tag{3.2}$$

Remark: In lieu of one initial value, r of them are selected for each processor. This will not create any burden. Since the result is for large time, the first a few values will not concern us. As was pointed out before, processor 1 begins its phase one operation at time 0, processor 2 starts at time 1, ..., processor r starts at time r-1.

We sketch the proof as follows. Without loss of generality, we may assume that $\{X_n^\varepsilon\}$ is bounded, otherwise, we can always use a truncation device as in [11]. For each N, define the truncation $x^{\varepsilon,N}(\cdot)$ by $x^{\varepsilon,N}(t) = x^\varepsilon(t)$ up until the first exit from the ball S_N. At the end, we take the limit as $N \to \infty$ to get the desired result. Due to the page limitation, we shall not pursue this matter here.

Since $\{X_n^\varepsilon\}$ is bounded, by (A1) $\{b(X_n, \xi_n)\}$ is uniformly integrable. By virtue of Lemma 3.7 in [11], $\{x^\varepsilon(\cdot)\}$ is tight in $D^r[0,\infty)$, where $D^r[0,\infty)$ denotes the space of functions that are right continuous and have left limits endowed with the Skorohod topology (cf. [11-13]). Extract a convergent subsequence, also indexed by ε (with no loss in generality) and with limit $x(\cdot)$. By Skorohod imbedding (cf. [11-13]), with a slight abuse of notation, we may assume that $x^\varepsilon(\cdot) \to x(\cdot)$ w.p.1. The convergence is uniform on any bounded time interval.

We thus need only characterize the limit process. To verify the limit $x(\cdot)$ satisfies (3.2), we show that $M_f(\cdot)$ defined by

$$M_f(t) = f(x(t)) - f(x(0)) - \int_0^t f'_x(x(u))\bar{b}(x(u))du \tag{3.3}$$

is a martingale for each smooth function $f \in C_0^2$, where C_0^2 stands for the space of C^2 functions with compact support.

Owing to the assumptions, $M_f(\cdot)$ is Lipschitz continuous. If $M_f(\cdot)$ is a martingale, then it must be a constant. Since $M_f(0) = 0$, $M_f(t) \equiv 0$ must hold. This in turn implies that $x(\cdot)$ solves a martingale problem with operator $\bar{b}'(x)\frac{\partial}{\partial x}$, or equivalently, $x(\cdot)$ satisfies (3.2).

To show that $M_f(\cdot)$ is a martingale, we need only prove the following assertion. For any bounded and continuous function $h(\cdot)$, arbitrary integer ν, and $t_j < t < t+s$ for $j \leq \nu$,

$$E(h(x(t_j)); j \leq \nu)\,(M_f(t+s) - M_f(t)) = 0. \tag{3.4}$$

To this end, we start with the sequence $\{x^\varepsilon(\cdot)\}$ and verify

$$\begin{aligned}&\lim_{\varepsilon \to 0} E\,(h(x^\varepsilon(t_j)); j \leq \nu)\,(f(x^\varepsilon(t+s)) - f(x^\varepsilon(t)))\\ &= E(h(x(t_j)); j \leq \nu)\left(\int_t^{t+s} f'_x(x(u))\bar{b}(x(u))du\right).\end{aligned} \tag{3.5}$$

Note that there is a sequence n_ε such that $n_\varepsilon \to \infty$ as $\varepsilon \to 0$ and $\varepsilon n_\varepsilon \to 0$. In what follows, we always assume that $t/(\varepsilon n_\varepsilon)$ and $s/(\varepsilon n_\varepsilon)$ are integers.

$$\begin{aligned} f(x^\varepsilon(t+s)) - f(x^\varepsilon(t)) &= \sum_{t \leq \varepsilon l n_\varepsilon < t+s} (f(X_{ln_\varepsilon + n_\varepsilon}) - f(X_{ln_\varepsilon}))\\ &= \varepsilon \sum_{t \leq \varepsilon l n_\varepsilon < t+s} f'_x(X_{ln_\varepsilon}) \sum_{ln_\varepsilon - d \leq j < ln_\varepsilon + n_\varepsilon - d} b(X_j, \xi_j).\end{aligned}$$

Consequently,

$$\begin{aligned}&\lim_{\varepsilon \to 0} E\,(h(x^\varepsilon(t_j)); j \leq \nu)\,(f(x^\varepsilon(t+s)) - f(x^\varepsilon(t)))\\ &= \lim_{\varepsilon \to 0} E\,(h(x^\varepsilon(t_j)); j \leq \nu)\\ &\quad \times \left(\sum_{\varepsilon l n_\varepsilon = t}^{t+s} (\varepsilon n_\varepsilon) \frac{1}{n_\varepsilon} \sum_{j = ln_\varepsilon - d}^{ln_\varepsilon + n_\varepsilon - d - 1} E^\varepsilon_{ln_\varepsilon - d} f'_x(X_{ln_\varepsilon}) b(X_j, \xi_j)\right).\end{aligned} \tag{3.6}$$

Since $d > 0$ is a fixed constant, $\varepsilon d \xrightarrow{\varepsilon} 0$. The continuity of $f_x(\cdot)$ and $b(\cdot, \xi)$ implies that the second sum in (3.6) can be replaced by

$$\sum_{j = ln_\varepsilon - d}^{ln_\varepsilon + n_\varepsilon - d - 1} E^\varepsilon_{ln_\varepsilon - d} f'_x(X_{ln_\varepsilon - d}) b(X_{ln_\varepsilon - d}, \xi_j).$$

In the above, we have used the weak convergence, Skorohod imbedding silently. The $E^\varepsilon_{ln_\varepsilon - d}$ was put in by the properties of conditional expectation and measurability of various terms with respect to the σ-algebra $\mathcal{F}^\varepsilon_{ln_\varepsilon - d}$. Let $\varepsilon l n_\varepsilon \to u$. Then, $\varepsilon(ln_\varepsilon - d) \xrightarrow{\varepsilon} u$. From the above paragraph, we need only show

$$\frac{1}{n_\varepsilon} \sum_{j = ln_\varepsilon - d}^{ln_\varepsilon + n_\varepsilon - d - 1} E^\varepsilon_{ln_\varepsilon - d} f'_x(x(u)) b(x(u), \xi_j) \xrightarrow{\varepsilon} f'_x(x(u))\bar{b}(x(u)) \text{ in probability.} \tag{3.7}$$

To do so, we approximate $x(u)$ by a simple process and then verify that (3.7) holds for each value of the simple process. To be more precise, for each $\delta > 0$, $x(u)$

can be approximated by a simple process $x^\delta(u)$, such that $|x(u) - x^\delta(u)| \le \delta$ and hence, the limit in probability of (3.7) is the same as that of

$$\frac{1}{n_\varepsilon} \sum_{j=ln_\varepsilon-d}^{ln_\varepsilon+n_\varepsilon-d-1} E^\varepsilon_{ln_\varepsilon-d} f'_x(x^\delta(u)) b(x^\delta(u), \xi_j).$$

Since $x^\delta(u)$ has only finitely many values, we can find a partition of the range of $x^\delta(u)$ into disjoint sets $B^\delta_1, \ldots, B^\delta_L$, such that $P\{x^\delta(u) \in \partial B^\delta_i\} = 0$, where ∂B^δ_i denote the boundary of B^δ_i. As a consequence, the limit in probability of (3.7) is the same as that of

$$\sum_{i=1}^{L} \frac{1}{n_\varepsilon} \sum_{j=ln_\varepsilon-d}^{ln_\varepsilon+n_\varepsilon-d-1} E^\varepsilon_{ln_\varepsilon-d} f'_x(x_i) b(x_i, \xi_j) I_{\{x^\delta(u) \in B^\delta_i\}}.$$

The desired result follows form this.

Remark: More complicated noise model, for example state dependent noise processes can be considered.

4. Some order of magnitude estimates

In view of (2.2), by virtue of Taylor expansion,

$$\begin{aligned} X_{n+1} - \theta &= X_n - \theta + \varepsilon b(X_{n-d}, \xi_{n-d}) \\ &= X_n - \theta + \varepsilon b_x(\theta, \xi_{n-d})(X_{n-d} - \theta) + \varepsilon b(\theta, \xi_{n-d}) + O(\varepsilon^2)(|b_{xx}(\varsigma, \xi_{n-d})| \, |X_n - \theta|^2) \end{aligned}$$

Therefore,

$$U^\varepsilon_{n+1} = U^\varepsilon_n + \varepsilon b_x(\theta, \xi_{n-d}) U^\varepsilon_{n-d} + \sqrt{\varepsilon} b(\theta, \xi_{n-d}) + O(\varepsilon\sqrt{\varepsilon})(|b_{xx}(\varsigma, \xi_{n-d})| \, |U_{n-d}|) \tag{4.1}$$

where $U^\varepsilon_n = \frac{X^\varepsilon_n - \theta}{\sqrt{\varepsilon}}$.

To get additional limit result, we assume further

(A3) $\bar{b}(x)$ has a unique stable point θ in the sense of Liapunov.

(A4) There is a twice continuously differentiable Liapunov function $V(\cdot) : \mathbb{R}^r \mapsto \mathbb{R}$ for (3.2), such that $V_{xx}(\cdot)$ is bounded uniformly, $|V_x(x)| \le K(1 + V(x))$, and

$$V(x) \ge 0, \; V(\theta) = 0$$

$$V(x) \to \infty, \text{ as } |x| \to \infty$$

$$V'_x(x)\bar{b}(x) < -\lambda V(x) \text{ for some } \lambda > 0.$$

(A5) $b(x, \xi) = b_1(x, \tilde{\xi}) + b_2(x)\hat{\xi}$, such that $\{\tilde{\xi}_n\}$ is a sequence of bounded random variables, $E\left(b_1(x, \tilde{\xi}_n) - \bar{b}(x)\right) = 0$ for each x and each n, and $\{\hat{\xi}_n\}$ is a sequence of i.i.d. zero mean random variables with $\sup_n E|\hat{\xi}_n|^2 < \infty$. For each $T < \infty$,

$$\sum_{j=n}^{T/\varepsilon} |E^\varepsilon_n V'_x(x)(b_1(x, \xi_j) - \bar{b}(x))| \le K(1 + V(x)) \text{ uniformly in } n.$$

Theorem 4.1. *Assuming (A1), (A3)-(A5) hold, and* $\{X_n^\varepsilon\}$ *is bounded. Then, the following order of magnitude estimate holds*

$$\limsup_{n\to\infty} EV(X_n^\varepsilon) = O(\varepsilon). \tag{4.2}$$

Remark: If the Liapunov function $V(\cdot)$ is locally quadratic, i.e., there is a positive definite matrix Q, such that

$$V(x) = (x-\theta)'Q(x-\theta) + o(|x-\theta|^2). \tag{4.3}$$

It is possible to show that there exist $0 < N_e \xrightarrow{\varepsilon} \infty$, such that

$$\{U_n^\varepsilon; n \geq N_\varepsilon\} \text{ is tight.} \tag{4.4}$$

Remark: The boundedness assumption on $\{X_n\}$ is not as restrictive as it may be seen. In practice, various projection or truncation methods are normally employed. These procedures in turn imply the boundedness. For notational simplicity, we choose to assume the boundedness condition.

We now indicate how Theorem 4.1 can be proved. The proof utilizes a technique known as perturbed Liapunov function methods developed in [11, 15]. Using a truncated Taylor expansion,

$$\begin{aligned} E_nV(X_{n+1}) - V(X_n) = {} & \varepsilon V_x'(X_n)\bar{b}(X_n) + \varepsilon V_x'(X_n)\left(\bar{b}(X_{n-d}) - \bar{b}(X_n)\right) \\ & + \varepsilon V_x'(X_n)\left(b_1(X_{n-d}, \xi_{n-d}) - \bar{b}(X_{n-d})\right) \\ & + \varepsilon V_x'(X_n) b_2(X_{n-d})\hat{\xi}_{n-d} \\ & + (X_{n+1} - X_n)'V_{xx}(\varsigma)(X_{n+1} - X_n) \end{aligned} \tag{4.5}$$

where ς is a random variable whose value is between X_n and X_{n+1}.

Observe that

$$X_{n-d} - X_n = -\varepsilon \sum_{j=n-2d}^{n-d-1} b_1(X_j, \xi_j) - \varepsilon \sum_{j=n-2d}^{n-d-1} b_2(X_j)\hat{\xi}_j. \tag{4.6}$$

It yields that $\bar{b}(X_{n-d}) - \bar{b}(X_n) = \bar{b}_x(\tilde{\varsigma})(X_{n-d} - X_n)$, where $\tilde{\varsigma}$ is a random variable whose value is between X_n and X_{n-d}. As a result,

$$\begin{aligned} \varepsilon V_x'(X_n)\left(\bar{b}(X_{n-d}) - \bar{b}(X_n)\right) = {} & -\varepsilon^2 V_x'(X_n)\bar{b}_x(\tilde{\varsigma}) \sum_{j=n-2d}^{n-d-1} b_1(X_j, \xi_j) \\ & - \varepsilon^2 V_x'(X_n)\bar{b}_x(\tilde{\varsigma}) \sum_{j=n-2d}^{n-d-1} b_2(X_j)\hat{\xi}_j. \end{aligned} \tag{4.7}$$

In view of (4.6) and the condition on $b_1(\cdot,\cdot)$, the first term on the right-hand side of (4.7) is bounded above by $O(\varepsilon^2)(1 + V(X_n))$.

By virtue of (A4) and (A5), it can be shown that the last term on the right-hand side of (4.5) is bounded by $O(\varepsilon^2)(1+V(X_n))+O(\varepsilon^2)|\hat{\xi}_{n-d}|^2$. Moreover, (A4) implies that $\bar{b}(x)$ also has linear growth. Now, introduce a perturbation of the form,

$$V_1^\varepsilon(n)=\varepsilon\sum_{j=n-d}^{T/\varepsilon-d} E_{n-d}^\varepsilon V_x'(X_n)\left(b_1(X_{n-d},\tilde{\xi}_j)-\bar{b}(X_{n-d})\right). \tag{4.8.1}$$

It is easily seen that

$$|V_1^\varepsilon(n)|\le O(\varepsilon)(1+V(X_n)), \tag{4.8.2}$$

i.e., the perturbation has a small magnitude. Furthermore, it results appropriate cancellations.

Define $V^\varepsilon(n)=V(X_n)+V_1^\varepsilon(n)$. By virtue of (4.5), (4.7) and various estimates,

$$\begin{aligned}E_n^\varepsilon V^\varepsilon(n+1)-V^\varepsilon(n)\le{}&-\varepsilon\lambda V(X_n)+\varepsilon V_x'(X_n)b_2(X_{n-d})\hat{\xi}_{n-d}\\&-\varepsilon^2V_x'(X_n)\bar{b}(\tilde{\varsigma})\sum_{j=n-2d}^{n-d-1}b_2(X_j)\hat{\xi}_j\\&+O(\varepsilon^2)(1+V(X_n))+O(\varepsilon^2)|\hat{\xi}_{n-d}|^2,\end{aligned}$$

Taking expectation and using (4.8), we get

$$EV^\varepsilon(n+1)-EV^\varepsilon(n)\le-\varepsilon\lambda EV^\varepsilon(n)+O(\varepsilon^2)(1+V^\varepsilon(n)).$$

The desired result follows from the above expression.

5. Further limit result

One of the interesting and important problem is to investigate the large time behavior of the algorithm. We wish to see what happens if $\varepsilon\to 0$, but $t\to\infty$. We have the following theorem.

Theorem 5.1. *Suppose the conditions of Theorem 3.1 together with (A3) hold, and the set $\{X_n^\varepsilon\}$ is tight. Suppose that $t_\varepsilon\to\infty$ as $\varepsilon\to 0$. Then, $x^\varepsilon(\cdot+t_\varepsilon)$ converges weakly to the unique stable point θ as $\varepsilon\to 0$.*

Remark: The above theorem indicates that if the algorithm is stable, then X_n^ε will be asymptotically close to θ. This kind of result is of special interest to various applications in estimation and tracking.

The proof of this theorem is in the essence of [11] and uses the stability condition in an essential way. We briefly discuss the argument involved here. Fix $T>0$ and consider a convergent subsequence $(x^\varepsilon(t_\varepsilon+\cdot),x^\varepsilon(t_\varepsilon-T+\cdot))$, with limit denoted by $(x(\cdot),x_T(\cdot))$. It is easily seen that $x(0)=x_T(T)$, and $x_T(0)$ belongs to a set which is tight. By virtue of the tightness and Theorem 3.1, for any $\eta>0$, there is a $T_\eta<\infty$ such that for all $T>T_\eta$,

$$P\{x_T(T)\in N(\eta,\theta)\}\ge 1-\eta,$$

where $N(\eta,\theta)$ denotes a η-neighborhood of θ. This yields $x(0)=\theta$ w.p.1, which is what we want to prove.

Another important problem is the rate of convergence issue. This rate result can be obtained by examining the limiting behavior of U_n^ε.

To start with, we take interpolation of U_n^ε. Define the interpolation $u^\varepsilon(\cdot)$ as follows:

$$u^\varepsilon(0) = U_{N_\varepsilon}$$

$$u^\varepsilon(t) = U_n^\varepsilon \text{ for } t \in [\varepsilon(n - N_\varepsilon), \varepsilon(n - N_\varepsilon + 1)). \tag{5.1}$$

Under appropriate conditions, $u^\varepsilon(\cdot)$ converges weakly to a diffusion process which satisfies a SDE.

Theorem 5.2. *Suppose (4.4) holds, and*

$$\sqrt{\varepsilon} \sum_{j=N_\varepsilon}^{t/\varepsilon-1} b(\theta, \xi_{j-d})$$

converges weakly to a Brownian motion with covariance $t\Sigma$. Suppose that $W(\cdot)$ is a standard Brownian motion and the following SDE

$$du = \bar{b}_x(\theta) u dt + \Sigma^{\frac{1}{2}} dW \tag{5.2}$$

has a unique solution (in the sense of distribution), $U_{N_\varepsilon} \Rightarrow u(0)$, and

$$\frac{1}{n} \sum_{j=m}^{m+n-1} E_m b_x(\theta, \xi_j) \xrightarrow{n} \bar{b}_x(\theta) \text{ in probability}$$

uniformly in m. Then, $\{u^\varepsilon(\cdot)\}$ is tight, and any convergent subsequence satisfies (5.2).

Remark: We have assumed the convergence to the Brownian motion process. Conditions guaranty such convergence can be derived. Normally, some kind of mixing conditions needs to be assumed. See the work of Billingsley [12], and Ethier and Kurtz [13] among others.

6. Concluding remarks

In this work, we proposed and analyzed a new recursive estimation model of stochastic approximation type. The essence is to utilize the pipeline structure. In lieu of using a single processor alone as in the classical approach, a collection of processors is employed. The processors are arranged as on a production line. They iterate on and update the same system. Recursive computations are executed cyclically and periodically. The newly computed value of each processor is passed to the next processor on line as soon as it is available. By concurrent using information and pipelining computation and communication, the algorithm can be implemented rather efficiently. One of the main advantages of this algorithm is that it can be done in real time.

By using the technique of weak convergence, we demonstrated that algorithms so designed preserves convergence and asymptotic normality. We have kept the step-size to be a constant throughout the paper. Similar result can be derived for a sequence of decreasing gain.

The problem of stopping rules can be considered. Such problems seem to be similar to the problem for the Robbins-Monro method [16].

Although the research in parallel estimation is relatively new, it has already shown promising signs. Further work in studying such algorithms and their implementations is needed. Design efficient and real time implementable algorithms is of particular importance.

References

[1] J. Tsitsiklis, Problem in decentralized decision making and computation, *Ph.D thesis*, Elect. Eng. Dept., MIT, Cambridge, MA, 1984.

[2] J.N. Tsitsiklis, D.P. Bertsekas and M. Athans, Distributed asynchronous deterministic and stochastic gradient optimization algorithms, *IEEE Trans. Automat. Control*, AC-31(1986), 803-812.

[3] H.J. Kushner and G. Yin, Asymptotic properties of distributed and communicating stochastic approximation algorithms, *SIAM J. Control Optim.*, 25(1987), 1266-1290.

[4] H.J. Kushner and G. Yin, Stochastic approximation algorithms for parallel and distributed processing, *Stochastics*, 22(1987), 219-250.

[5] G. Yin and Y.M. Zhu, On w.p.1 convergence of a parallel stochastic approximation algorithm, *Probab. Eng. Inform. Sci.*, 3(1989), 55-75.

[6] G. Yin and Y.M. Zhu, On robustness of the Robbins-Monro method for parallel processing, *System Control Lett.*, 12(1989), 77-86.

[7] Y.M. Zhu and G. Yin, Stochastic approximation in real time: pipelining of computation and communication, preprint.

[8] G. Yin and Y.M. Zhu, On parallel processing approach to adaptive stochastic approximation, *Proc. 27th Conf. Decision Control.*

[9] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computing*, Prentice Hall, New Jersey, 1989.

[10] M.J. Quinn, *Designing Efficient Algorithms for Parallel Computers*, McGraw-Hill, New York, 1987.

[11] H.J. Kushner, *Approximation and Weak Convergence Methods for Random Processes with Applications to Stochastic Systems Theory*, MIT Press, Cambridge, MA, 1984.

[12] P. Billingsley, *Convergence of Probability Measures*, J. Wiley & Sons, New York, 1968.

[13] S. Ethier and T.G. Kurtz, *Markov Processes*, J. Wiley & Sons, New York, 1986.

[14] H.J. Kushner and A. Shwartz, An invariant measure approach to the convergence of stochastic approximations with state dependent noise, *SIAM J. Control Optim.*, 22(1984), 13-27.

[15] H.J. Kushner and H. Huang, Asymptotic properties of stochastic approximations with constant coefficients, *SIAM J. Control Optim.*, 19(1981), 87-105.

[16] G. Yin, A stopping rule for the Robbins-Monro method, to appear in *J. Optim. Theory Appl.*, 1989.

NOMINAL ADAPTIVE CONTROL FOR A ROBOT MANIPULATOR

D. M. Dawson and J. F. Dorsey
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

This paper derives an adaptive controller to achieve trajectory following for a robot manipulator. Lyapunov's second method is used for the design. The parameter, position, and velocity errors are shown to be uniformly bounded in the presence of an unknown bounded disturbance.

I. INTRODUCTION

In this paper, we propose a controller that achieves trajectory following without requiring exact knowledge of the nonlinear dynamics described by

$$f = M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + F_d\dot{q} + F_s(\dot{q}) + Td \qquad (1.1)$$

where $M(q)$ is an nxn inertia matrix (bounded, symmetric, and positive definite), $V(q,\dot{q})$ is an nxn matrix containing the centrifugal and coriolis terms, $G(q)$ is an nx1 vector containing the gravity terms, F_d is a constant diagonal nxn matrix containing the dynamic coefficients of friction, $F_s(\dot{q})$ is an nx1 vector containing the static friction terms, Td is an nx1 vector representing an unknown, bounded disturbance, $q(t)$ is an nx1 joint variable vector, and f is an nx1 control input vector.

In this paper, we are concerned with adaptive controllers that do not require knowledge of the robot parameters such as the link masses or the coefficients of friction. In [4], Craig employs an adaptive control law to achieve trajectory following without knowledge of the robot parameters. Specifically, Craig's control law restricts the parameter estimates to lie within known bounds. If a parameter estimate moves outside its known bound, it is reset to within its known bound.

Instead of resetting the parameters, our design procedure is to chose an upper and lower bound and a nominal midrange value for that parameter. Specifically, we change the standard adaptive control scheme by augmenting the adaptive update law to include a nominal parameter value term. The advantage of this method is that we have included any unknown bounded disturbances in our initial analysis; therefore, we do not have to redesign our controller due to robustness considerations.

II. NOMINAL ADAPTIVE CONTROL

For the adaptive schemes considered in the literature, the presence of a bounded disturbance can result in an unbounded parameter estimate. With a bounded disturbance Td in (1.1), it is easy to show in the Lyapunov approach that $\dot{V} < 0$ if

$$\|\underline{e}\| > \frac{\|2PB\| \; \|Td\|}{\lambda\min\{Q\}}.$$

Therefore, it is clear that the tracking error will remain bounded; however, the parameter error may diverge.

Our control and update law are shown below:

Control: $f_c = W\hat{\phi} + K_v\dot{e} + K_pe$

Update: $\dot{\hat{\phi}} = \eta W^T(\alpha e + \dot{e}) - \eta\beta\hat{\phi} + \eta\beta\phi_o$

where β represents a positive scalar control gain to be adjusted, K_p, K_v, W are as defined in [3], and ϕ_o is an rx1 vector containing the nominal values for the parameters.

With the above update law, a bound can be assigned to the parameter and tracking errors in terms of the various control parameters.

$$\left\| \begin{bmatrix} e \\ \dot{e} \\ \tilde{\phi} \end{bmatrix} \right\| \leq \sqrt{\frac{(2\alpha k_v + \alpha m_2(1 + \alpha))}{m_1(1 - \alpha)}} \left\| \begin{bmatrix} \alpha Td/\beta \\ Td/\beta \\ \phi_d \end{bmatrix} \right\|$$

As discovered by various simulation runs, the above bound tends to be very conservative. The main adavantage is that we have ensured that the parameter error remains bounded.

This turns out to be bandaid approach (i.e. we have sacrificed performance to ensure stability. The bounded disturbance problem led us to examine the possibilities of robust control.

III. CONCLUSION

In summary, we have changed the standard adaptive control scheme by augmenting the adaptive update law to include a nominal parameter value term. The parameter, position, and velocity errors are shown to be uniformly bounded in the presence of an unknown bounded disturbance. We show that the error and its derivative are bounded within a neighborhood of zero with a radius that is a function of the magnitude of known and unknown disturbances.

References are available upon request.

REAL-TIME VISION SERVOING OF A MANIPULATOR WITH SELF-TUNING CONTROLLER

N. HOUSHANGI and A.J. KOIVO
Electrical Engineering Department, Purdue University, West Lafayette, IN 47907

INTRODUCTION

An adaptive self-tuning controller in conjunction with visual information has been used in [1] to pick up a stationary object. The information received is extracted before manipulator motion begins, i.e., image processing and manipulator control are not interacting during motion. The work addressed here is to grasp a moving object by adaptively controlling a robot manipulator using on-line vision feedback. The adaptive controller has the ability to adapt to unknown payloads without deteriorating its performance.

PREDICTION OF OBJECT POSITION

The position $p^o(kT_v)$ of the object is described by

$$p^o(kT_v) = [p_x^o(kT_v)\ p_y^o(kT_v)\ p_z^o(kT_v)]^T \tag{1}$$

where k=0,1,..., T_v is the vision sampling period, subscripts x, y, z refer to the coordinate axes of the Cartesian base coordinate system, and superscript T indicates transposition.

The object is assumed to exhibit a smooth motion, i.e., position and velocity are continuous. In addition, it is circumscribed by a sphere and has a maximum velocity less than the maximum velocity of the gripper. Object motion will be described by specifying the position of the center of the sphere versus time. Determination of object position from camera images is a time consuming process, which causes a delay in position information, preventing its use in on-line control. Therefore, the position is predicted one vision sampling period ahead [2]. It is also necessary to provide the controller with the position of the object at the sampling instants (kT_v+mT_c) of the control algorithm, where T_c is the sampling period in the control scheme and $m=1,\ldots,T_v/T_c$ is an integer.

The foregoing prediction is based on velocity values, which are calculated from the successive positions of the object determined from recorded images. The velocity of the object is calculated by the following expression :

$$v^o(kT_v) = \{p^o(kT_v) - p^o[(k-1)T_v]/T_v\} \tag{2}$$

where $v^o(kT_v) = [v_x^o(kT_v)\ ,\ v_y^o(kT_v)\ ,\ v_z^o(kT_v)]^T$ is the velocity of the object at time kT_v. The velocity determined by equation (2) is accurate if T_v is sufficiently small.

The velocity values calculated by equation (2) are used next to construct an auto-regressive (AR) discrete time model to predict future values of object velocity. This model is chosen as :

$$v^o(kT_v) = A^o(z^{-1})v^o(kT_v) + e^o(kT_v) \tag{3}$$

The matrix polynomial $A^o(z^{-1})$ is specified by:

$$A^o(z^{-1}) = A_1^o z^{-1} + A_2^o z^{-2} + \cdots + A_n^o z^{-n} \tag{4}$$

where the (3×3) matrix ,A_i^o, i = 1,...,n, contains unknown parameters: z^{-1} being a delay operator relative to the vision sampling period, i.e. $z^{-1}v^o(kT_v) = v^o[(k-1)T_v]$; and n specifying the order of the AR-model. The unknown parameters in matrix A^o are first estimated using the least square estimation, and then velocity values are predicted. These values determine a position of the object at a specified future time. To estimate the unknown parameter values in equation (3), the following matrices are defined:

$$M^o(k-1) = [(v^o(k-1))^T, \ldots, (v^o(k-n))^T]^T \tag{5}$$

$$\alpha^o = [A_1^o, \ldots, A_n^o]^T = [\alpha_1^o, \ldots, \alpha_3^o] \tag{6}$$

where the vision sampling period T_v has been dropped in the arguments. Using the foregoing notations, equation (3) is rewritten as :

$$v^o(k) = (\alpha^o)^T M^o(k-1) + e^o(k) \tag{7}$$

The least square error method [3] is used to estimate parameter vector α^o in equation (7).

The parameter estimates calculated on-line are then substituted into equation (3). The resulting model gives the predicted object velocity. These values are used to determine the object position. Specifically, the velocity $v^o(k+r|k-1)$ of the object is predicted r steps ahead by (7), when past measurements of v^o up to and including time $(k-1)T_v$ are given. Thus, the predicted velocity value is

$$\hat{v}^o(k+r|k-1) = [\hat{\alpha}^o(k)]^T \hat{M}^o(k-1+r|k-1) \tag{8}$$

$$\hat{M}^o(k-1+r|k-1) = [(\hat{v}^o(k-1+r|k-1))^T,\ldots.,(\hat{v}^o(k|k-1)^T),(v^o(k-1)^T,\ldots.,(v^o(k+r-n))^T]^T \tag{9}$$

The position of the object is needed at the sampling instances $[(k+r)T_v+mT_c)$ for the control scheme, where integer $m = 1,\ldots.,T_v/T_c$. By assuming a constant velocity over sampling period $[(k+r-1)T_v,(k+r)T_v)$, the predicted point is calculated as follows:

$$\hat{p}^o[(k+r)T_v+mT_c\,|\,(k-1)T_v] = \hat{p}^o[(k+r)T_v\,|\,(k-1)T_v]+\hat{v}^o[(k+r)T_v\,|\,(k-1)T_v]mT_c \tag{10}$$

EXPERIMENTAL RESULTS

The practicality of the approach is demonstrated by a lab experiment in which a moving object is grasped using vision information in conjunction with a self-tuning controller. For the experiment, the commercially available industrial robot, Puma 600 (Unimation), is used. A cylindrical object standing on its base moving on the xy-plane is to be grasped by the end-effector of the Puma manipulator. It is assumed that the height of the object is known. In order to grasp the object, the position of the center point of the gripper should be made to coincide with the center of gravity of the object. An image is grabbed at the start of each vision sampling period with a CCD camera mounted on the ceiling 1.5m above the robot workspace. The size of the image taken is 512×512 pixels with 256 gray levels. The geometric center and radius of the object are calculated from the computer images.

The system implemented represents an adaptive vision feedback system. The motion of the object is modeled by means of a second order AR-model. This model gives the predicted velocity and position of the object so as to compensate for the time delay caused by the imaging system. The predicted object position values are used in a planning algorithm, which generates on-line the desired (nominal) trajectory point for the motion of the hand. The nominal trajectory values generated for the end-effector are used in the adaptive self-tuning control scheme so that the end-effector will track the moving object. When the system converges, the gripper ultimately is able to grasp the moving object. A typical graph of an experiment is shown in figure 1. The described system will track the changing object position by modifying the desired trajectory of the manipulator on-line. Thus, the system exhibits intelligence in the aforementioned sense.

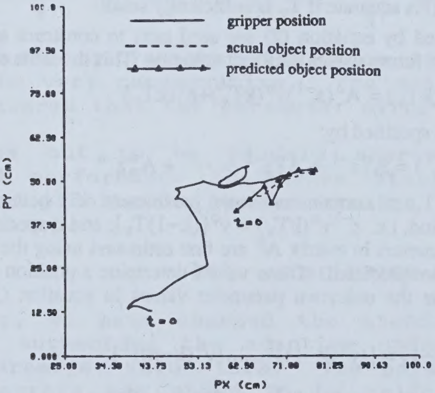


Figure 1: Experimental gripper position versus actual and predicted object position

References

1. T. H. Chiu, A. J. Koivo, and R. Lewczyk, "Experiments on Manipulator Gross Motion Using Self-tuning Controller and Visual Information," *Journal of Robotic Systems*, vol. 3(1), 1986.
2. Y. P. Chien, *Ph.D. Thesis*, Purdue University, August 1989.
3. A. J. Koivo, *Fundamentals of Control of Robot Manipulators*, J. Wiley & Sons, NY, 1989.

A COMPARISON OF ARMA AND IMPULSE IDENTIFICATION

Mark S. Moellenhoff and Kelvin T. Erickson
Department of Electrical Engineering
University of Missouri-Rolla
Rolla, Missouri 65401

ABSTRACT

This paper compares the identification of step response model coefficients based on either an autoregressive moving average or an impulse response model. The identification of the two models is compared for speed and accuracy in achieving the ideal step response coefficients. This comparison is useful for those wishing to construct an adaptive form of the model-based predictive controllers that use an internal step response process model.

INTRODUCTION

Development of digital model-based predictive controllers such as Dynamic Matrix Control [1], Model Algorithmic Control [2][3], Internal Model Control [4] and Forward Model Controller [5] were due to limited success of modern control theory on chemical processes. An impulse or step response model is used by these control schemes to predict future trends of the process outputs and manipulate process inputs to achieve the desired process outputs. The predictive controllers themselves may need to be updated in order to accurately control processes with characteristics that change significantly. The mentioned controllers become adaptive merely by identifying the process models on-line. This study compares the performance of the ARMA and impulse identification when used to obtain the step response models needed for these control algorithms

Both autoregressive moving average and impulse identification methods can identify the process models on-line. However, the step response coefficients are not identified directly as needed. Coefficients of difference equation or impulse response models are identified instead. From both of these models, the step response model can be obtained. The polynomial formed by the ARMA model difference equation is divided to obtain impulse coefficients. Impulse coefficients are then integrated (or summed) to obtain the step response coefficients. Both methods must ultimately obtain the step response coefficients since they are used in the comparison.

With the identified step response coefficients, a process engineer can construct an adaptive control system. The identification model chosen is based on specification needs. The comparisons following will help determine which identification model the user will choose.

The remainder of the paper is organized as follows. General information about the identifiers and experiment is first presented followed by information more specific to this experiment. Results of the experiment are then tabulated followed by pertinent conclusions.

BACKGROUND INFORMATION

The system used for this experiment is shown graphically in Fig. 1. The noise into the system is modeled as a quantity added to the output of the process, resulting in corruption of the output data. Both the input and corrupted output values are used by the identifier to identify the coefficients of the process model. The types of coefficients identified are different due to the different identification methods used. In spite of the different coefficients, a fair comparison must be followed.

For this study, the general ARMA model is given by:

$$\frac{Y(z)}{X(z)} = \frac{a_1 z^{-1} + a_2 z^{-2} + \ . \ . \ . + a_n z^{-n}}{1 + b_1 z^{-1} + b_2 z^{-2} + \ . \ . \ . + b_n z^{-n}}$$

The general impulse model is assumed to be:

$$\frac{Y(z)}{X(z)} = h_1 z^{-1} + h_2 z^{-2} + \ . \ . \ . + h_n z^{-n}$$

Both identification models have advantages and disadvantages. The difference equation is essentially a ratio of two polynomials. This ratio must be divided to obtain its equivalent impulse coefficients, thus requiring more computation. The advantage of the ratio is the possiblility of needing only a finite number of ARMA coefficients to obtain an infinite number of impulse coefficients. The impulse identifier, on the other hand, must identify more coefficients than the difference equation identifier since each impulse coefficient must be identified. Since with the impulse identifier an infinite number of coefficients can theoretically be identified, a priori information about model order, or time delay is not necessary. Unusual process dynamics that require special information about the process being identified in other models are handled naturally in the impulse model. Once the impulse model is obtained using either method, a step response model can be obtained by integrating the impulse model. Step response models as a whole have good intuitive appeal to process operators.

For each of the models, the U-D factorization of the recursive least squares (RLS) algorithm is used for identification. This method was chosen due to its popularity and convergence properties [6]. The particular implementation of U-D factorization was taken from [7].

To successfully identify the coefficients of the process, the input and output data must be "sufficiently rich" in information [6]. A pseudo-random binary sequence (PRBS) is used as a test signal injected into the process to excite the significant dynamics of the process.

Noise is added to the output of the process to obtain a specific signal-to-noise ratio (SNR). SNR is, by definition, the ratio of the test signal variance to the noise variance. For a specific PRBS test signal amplitude, a_t , the variance is a_t^2. Since the SNR is determined at the process output, the variance of the test signal must be transferred through the process. According to [8], for a linear combination of random variables:

$$y = A_1 x_1 + A_2 x_2 + A_3 x_3$$
$$\mathrm{Var}(y) = A_1^2 * \mathrm{Var}(x_1) + A_2^2 * \mathrm{Var}(x_1) + A_3^2 * \mathrm{Var}(x_3)$$

where x_1 , x_2 , and x_3 are random variables. Therefore,

$$\begin{aligned}\mathrm{Var}(\text{test sig. @ process out}) &= \mathrm{Var}(\text{test sig. @ process in}) * \sum_{i=1}^{n} h_i^2 \\ &= a_t^2 * \sum_{i=1}^{n} h_i^2\end{aligned}$$

As will be explained below, for variance purposes, the noise can be treated as a continuous variable. As the method is explained in [8], the variance for the noise equals $\frac{a_n}{3}$.

The random number generator was obtained from [9]. Given an initial seed value, the random number generator determines a random number between 1 and 729. In the case of the noise signal, 1 through 729 was mapped into [$-a_t$, $+a_t$]. Since the noise magnitude is typically a small number (i.e. between .5 and 3.5), the noise can be approximated as a continuous variable when determining its variance. In the case of the PRBS test signal, numbers between [1, 364] are assigned a value of $-a_t$, and numbers between [365, 729] are assigned the value

$+a_t$. The formulas for the variance of the noise signal through the process were verified numerically with the random number generator and process used in this experiment.

COMPARISONS

The speed and accuracy of the ARMA and impulse model identification are compared for changes in the process gain and deadtime with various SNR at the process output. For comparison under realistic conditions, three values of SNR are utilized at the process output. Regardless of the type of model to be identified, the step response coefficients are used for comparison. To obtain the step response coefficients from the ARMA model parameters, the numerator is divided by the denominator to obtain the impulse response coefficients. The step coefficients are related to the impulse coefficients by the equation,

$$a_i = a_{i-1} + h_i = \sum_{k=1}^{i} h_k.$$

The speed of the model identification is an indication of the ability to adapt to changing process conditions. For this study, the identifier speed is defined as the number of sample periods before the step response of the identifier model falls within a certain tolerance (±20%) of the goal process step response.The ±20% tolerance is computed for the last step response model coefficient, and is used to compare against all 20 coefficients. This results in a constant tolerance range whose width is based solely on the final coefficient as shown in Fig. 2.

The model accuracy for this experiment is composed of two quantities; namely, steady-state accuracy and model fluctuation. The steady-state accuracy indicates the closeness of the step response of the identified model to the step response of the goal process for a large number of samples. The steady-state accuracy, as determined in this experiment, is the deviation from the actual coefficient as a percentage of the last coefficient, at the 1000^{th} sample.

$$\text{percent}_i = \frac{\text{coeff}_i - \text{actual\_coeff}_i}{\text{coeff}_{20}} * 20\%$$

where coeff_i is the i^{th} identified coefficient and actual_coeff_i is the i^{th} goal process coefficient.

Each model contains 20 coefficients and thus 20 percentages which represents the steady-state accuracy. All 20 percentages cannot be used. The value representing the steady-state accuracy is the maximum percent of all 20 coefficients computed for each model. The fluctuation, on the other hand, refers to the degree of oscillation the identified step coefficients acheive at the 1000^{th} sample. For this experiment, the degree of fluctuation is determined by the number of samples for which the step response model has been within the ±20% tolerance range at the 1000^{th} sample.

These two comparisons: speed of adaptation, and model accuracy, respectively, will aid the control engineer in determining which identification technique to employ. In control terms, the speed and steady-state accuracy are similar to settling time and steady-state accuracy of a transient response. The fluctuation helps determine the degree to which the step response coefficients will stay within the tolerance band after a large number of samples. The control engineer must decide which comparison best parallels his/her needs; namely fast response time or accuracy.

EXPERIMENTAL DETAILS

A standard procedure is followed in the experiment to obtain a fair comparison between identification methods. An initial process model is identified to obtain desired initial conditions in the identifier. The process is changed to a common goal process, the input signal and SNR are chosen, and the goal process is identified. While the goal process is being identified, data is collected. This experimental procedure thus allows the comparison of the identification methods as the coefficients change from an initial model to a goal model.

For this paper, the process is a first-order process with deadtime,

$$G(s) = \frac{Ke^{-dTs}}{\tau s + 1}$$

where K is the gain of the process, d is the deadtime, τ is the time constant, and T is the sampling period. For these tests, the sample period is one second. To simulate the digital control of the process, the discrete equivalent of the process preceeded by a zero-order hold is represented as

$$G(z) = K\,\frac{(1 - e^{-T/\tau})z^{-d}}{1 - e^{-T/\tau}z^{-1}}$$

The common goal process has the parameters,

$K = 1$

$d = 2$ sec

$\tau = 2$ sec

When comparing identification methods for changes in process gain, the initial process has the following parameters:

$K = 0.5$ or 2.0
$d = 2$ sec
$\tau = 2$ sec

When comparing methods for changes in process deadtime, the initial process has the following parameters:

$K = 1$
$d = 1$ or 4 sec
$\tau = 2$ sec

The identification methods are compared for open-loop and closed-loop conditions. To test the performance under open-loop conditions, the controller is in manual mode, i.e., holds the controller output at a constant value. The PRBS signal needed for identification is added to this controller output value. For closed-loop conditions, the controller implements the model-based predictive control algorithm described by Otto [5]. For these tests, the main tuning parameter, the closed loop settling time is set to 10 seconds, which corresponds to mild control action. Under closed- loop conditions, the PRBS needed for identification is added to the controller output.

For this experiment, the number of parameters identified in the ARMA process was constant for all initial models. With this constraint, the number of ARMA coefficients identified had to correspond to the maximum possible amount of deadtime used in an initial model. Since the maximum amount of deadtime was 4 sample periods and the process known to be first order, 5 numerator and 1 denominator coefficients were identified.

A common sequence of process changes and identifications was followed by both identifiers for fair comparisons.

1. Set the process to a specific initial process.
2. Set the internal identifier model coefficients, assuming perfect identification, to the desired initial process.
3. Freeze the identification routine at the current state.
4. Change the process to the common goal process.
5. Initialize the identifier.
    (Identified coefficients left intact)
    (Diagonal D matrix initialized to a large quantity)
    (Offdiagonal U matrix initialized to zero)
6. Set the PRBS signal to the desired value and set the process noise

magnitude to get a certain SNR.

7. Unfreeze the identification routine.
8. Identify coefficients while collecting data.

A PRBS test signal magnitude of ±4 was used for the process input. For the following SNR's, the noise amplitude was as follows:

SNR	Noise Amplitude (maximum)
1	3.422
2	2.420
4	1.711

EXPERIMENTAL RESULTS

The data obtained using the techniques mentioned above is summarized in Table 1 for open-loop conditions and in Table 2 for closed-loop conditions. The different columns in the tables represent:

1st Occurence:	number of data samples before the step response coefficients are within 20% tolerance.
10th Occurence:	number of data samples before the step response coefficients are within 20% tolerance for 10 consecutive samples.
20th Occurence:	number of data samples before the step response coefficients are within 20% tolerance for 20 consecutive samples.
Final Count:	The number of samples for which the step response model has been within the 20% tolerance range at the 1000th sample.
Max Deviation:	The maximum percentage out of tolerance for the 20 step response coefficients at the 1000th sample.

Common control terms such as rise time and steady-state accuracy can be used to make comparisons between the ARMA and Impulse identification data. The results listed in Table 1 suggests the following comparisons for the two identification methods under open-loop conditions:

1. The ARMA model has a faster rise time in obtaining step response

coefficients within the ±20% range.

2. The step response model obtained from the impulse model has greater steady-state accuracy.
3. Use of the impulse model for identification leads to less fluctuation of the step response coefficients within the 20% tolerance range.

CONCLUSIONS

In this paper, we have compared the identification of the step response coefficients for an autoregressive moving average versus an impulse model. The identified step response models were compared for the speed and accuracy of the model. Speed refers to the ability of the model to adapt to changing process conditions. The model accuracy, comprised of steady-state accuracy and model fluctuation, represented the closeness and fluctuation of the identified step response coefficients as compared to the ideal step response coefficients.

The ARMA model is less accurate, has more fluctuation, but is faster in obtaining ideal step response coefficients. The impulse model, on the other hand, is more accurate, has less fluctuation, but is slower in obtaining the ideal step response coefficients.

BIBLIOGRAPHY

[1] C. R. Cutler and B. L. Ramaker, "Dynamic Matrix Control - A computer control algorithm", *Proc ACC*, paper WP5-B, 1980.

[2] J. Richalet, A. Rault, J. L. Testud, J. Papon, "Model predictive heuristic control: applications to industrial processes", *Automatica*, vol. 14, 413-428, 1978.

[3] R. Rouhani and R. K. Mehra, "Model Algorithmic Control; basic theoretical properties", *Automatica*, vol. 18, 401-414, 1982.

[4] C. E. Garcia and M. Morari, "Internal Model Control: A unifying review and some new results", *Ind. Eng. Chem. Process Des. Dev.*, vol 21, 308-323, 1982.

[5] R. E. Otto, "Forward Modelling Controllers: A comprehensive SISO controller", AIChE Annual Meeting, 1986.

[6] D. E. Seborg, T. F. Edgar, S. L. Shah, "Adaptive Control Strategies for Process Control: A Survey", *AIChe Journal*, Vol. 32, 881-913, 1986.

[7] K. J. Åström and B. Wittenmark, *Computer Controlled Systems*, Prentice - Hall Inc., 324-342, 1984.

[8] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, Second Edition, Brooks/Cole Publishing (1987).

[9] W. J. Savitch, *Pascal: An Introduction to the Art and Science of Programming*, Benjamin/Cummings Pub. Co., page 244 , (1984).

[10] J. D. Morningred, D. A. Mellichamp, D. E. Seborg, "On-Line Adaptation of Predictive Controllers ", *Proc ACC*, vol 3., 1731-1736, 1987.

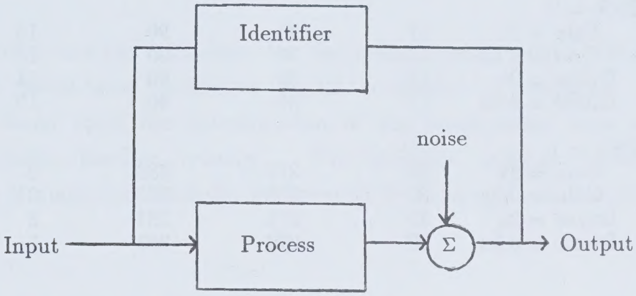


Figure 1. Experimental System

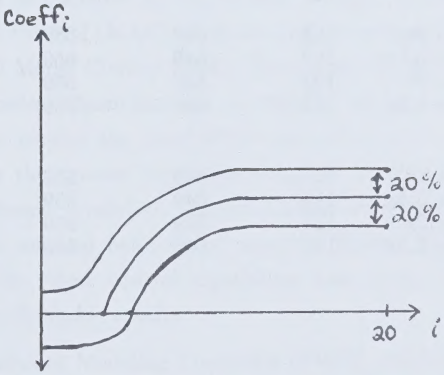


Figure 2. Tolerance Range

Table 1. Identification results for open-loop conditions.

Description	1st Occurence	10th Occurence	20th Occurence	Final Count	Max Deviation
ARMA Ident					
SNR = 4					
Gain = 2x	12	28	38	23	18%
Gain = 0.5x	12	28	38	21	17%
Dtime = 2x	12	28	38	23	18%
Dtime = 0.5x	12	28	38	23	17%
SNR = 2					
Gain = 2x	17	36	90	15	20%
Gain = 0.5x	17	36	90	16	19%
Dtime = 2x	17	36	90	14	20%
Dtime = 0.5x	17	36	90	15	19%
SNR = 1					
Gain = 2x	32	273	283	2	19%
Gain = 0.5x	32	273	283	0	21%
Dtime = 2x	32	273	283	2	19%
Dtime = 0.5x	32	273	283	1	20%
Impulse Ident					
SNR = 4					
Gain = 2x	103	248	652	373	10%
Gain = 0.5x	103	248	652	376	12%
Dtime = 2x	103	248	652	373	8%
Dtime = 0.5x	103	248	652	371	7%
SNR = 2					
Gain = 2x	103	649	659	235	15%
Gain = 0.5x	103	649	659	232	10%
Dtime = 2x	103	649	659	233	11%
Dtime = 0.5x	103	649	659	232	10%
SNR = 1					
Gain = 2x	113	649	659	4	16%
Gain = 0.5x	113	649	659	0	17%
Dtime = 2x	113	649	659	2	15%
Dtime = 0.5x	113	649	659	4	20%

STABILITY ANALYSIS OF A MULTIVARIABLE MODEL-BASED PREDICTIVE CONTROLLER

ANTHONY D. RANEY and KELVIN T. ERICKSON
Department of Electrical Engineering
University of Missouri-Rolla
Rolla, Missouri 65401

ABSTRACT

A stability analysis technique for the multivariable Forward Modeling Controller, a model-based predictive digital controller, is presented. The technique is based upon the determination of the transmission zeros of the controller transfer function matrix. The stability analysis allows the determination of maximum controller performance for both square and non-square systems.

I. INTRODUCTION

The limited successful application of modern control theory to chemical process control motivated the development of digital model-based predictive control algorithms such as Dynamic Matrix Control (DMC) (Cutler and Ramaker [1]), Model Algorithmic Control (MAC) (Richalet and co-workers [2], Rouhani and Mehra [3]) and Internal Model Control (IMC) (Garcia and Morari [4], [5]). These multivariable digital control algorithms use an impulse response or step response model of the process to predict the trend of the process outputs and to compute the required change in the process inputs to bring the outputs to their desired values. These model-based predictive control schemes were formulated to deal with the deadtime and unusual behavior of complex MIMO chemical industrial processes. Consequently, these control algorithms have been applied to many chemical processes with favorable results.

Recently, the Forward Modeling Controller (FMC), which circumvents the difficulties of the other model-based predictive controllers was developed for use on single input, single output (SISO) processes by Otto [6]. The basic FMC algorithm is extended for use on multivariable processes in Erickson and Otto [7]. The multivariable FMC algorithm is conceptually similar to other model-based predictive digital controllers but is a computationally simpler algorithm. Unlike the DMC or the MAC, the multivariable FMC does not require the on-line solution of an optimization problem. In contrast to the MAC, the FMC does not require the solution of an off-line Ricatti difference equation for processes with nonminimum phase. The multivariable FMC also does not require the derivation of an approximate process inverse as in the IMC. The response models are used directly in the calculation of the controller outputs.

This paper develops a technique for determining system stability for a given set of controller tuning parameters whether the system is square or non-square. The stability analysis of a non-square system is important even for systems which are normally square. Loss of a sensor or actuator may cause system instability. We would like to be able to predict such instability and adjust the controller tuning parameters to avoid instability.

The remainder of the paper is organized as follows. The transfer function form of the controller is developed in the next section. In section III, the method for determining system stability is outlined. In section IV, an ethanol/water distillation column simulation is used as an example non-square system for determining stability over the range of controller tuning parameters.

II. CONTROLLER TRANSFER FUNCTION

In the multivariable FMC, the process has q outputs and r inputs and is modeled by a discrete impulse response model:

$$y_i = H_1 m_{i-1} + H_2 m_{i-2} + \cdots + H_N m_{i-N} + d_i \qquad (1)$$

where

$y_i = [\ y_1\ \ y_2\ \ \cdots\ \ y_q\ ]^T$, a q x 1 vector of process output measurements,

$m_i = [\ m_1\ \ m_2\ \ \cdots\ \ m_r\ ]^T$, an r x 1 vector of manipulated variables,

H_k is a q x r matrix of the k^{th} impulse response coefficient; $(h_{ij})_k$ is the k^{th} impulse response coefficient of the response between the j^{th} manipulated variable and the i^{th} process output,

$y_i = [\ y_1\ \ y_2\ \ \cdots\ \ y_q\ ]^T$, a q x 1 vector of the current discrepancies between the model and the measurement.

An impulse response model of the process is also employed by others (Cutler and Ramaker [1]; Marchetti and co-workers [8]; Richalet and co-workers [2]; Rouhani and Mehra [3]), and has the significant advantages:

- No *a priori* assumptions about model order, time delay, etc. are necessary.

- Unusual process dynamics are handled naturally and do not require the specification of model structure.
- The coefficients of the model can be obtained from simple step response data.
- The step response model, obtained by integrating the impulse response model has high intuitive appeal to process operators.
- With prediction error identification methods, multivariable MA models will, in the limit, converge to the true parameter values (Stoica and Söderström [9]).

The chief disadvantages of an impulse response model are

- Non-mimimality of the representation, i.e., the relatively larger number of parameters used in the model, compared with the smaller number of parameters associated with a low-order transfer function model.
- An *a priori* assumption of the time to steady state.

However, we feel the advantages of an impulse response model far outweigh the disadvantages.

The multivariable FMC computes a prediction of the process output, T_j, into the future with the following assumptions:

1. The manipulated variable is held constant into the future
   $m_i = m_{i+1} = m_{i+2} = \cdots,$
2. The future discrepancies remain at their current value
   $d_i = d_{i+1} = d_{i+2} = \cdots,.$

The controller calculates an output m_i which, if it were kept constant from here on, would minimize the Euclidean norm of the error between the prediction and the setpoint at P sample intervals in the future. In conventional controller terms (Fig. 1), the controller transfer function can be shown to be [7],

$$C(z) = \left[(A_P + H_{P+1}z^{-1} + \cdots + H_N z^{-N+P}) - (H_1 z^{-1} + \cdots + H_N z^{-N})\right]^{(-1)}. \qquad (2)$$

where $A_j = \sum_{k=1}^{j} H_k$ is the j^{th} "matrix coefficient" of the process step response.

The notation $X^{(-1)}$ indicates the pseudo-inverse of a non-square matrix. Of course, if X is square, then the pseudo-inverse is the usual matrix inverse.

The output of the conventional closed-loop system is expressed as

$$y(z) = \left[I + G(z)C(z)\right]^{(-1)} G(z)C(z)s(z). \qquad (3)$$

One way to examine stability of the closed-loop system is to examine the system poles. Obviously, not a trivial task for (3). However, if we convert the controller to the Internal Model configuration (Fig. 2), stability analysis is simpler. Garcia and Morari [5] show that the output of the system in Fig. 2 is given by

$$y(z) = G(z)\Big[I + G_c(z)\big(G(z) - G_m(z)\big)\Big]^{(-1)} G_c(z)\big(s(z) - d(z)\big) + d(z). \tag{4}$$

Now, if exact process modeling is assumed, then G_m is set to G and (4) becomes

$$y(z) = G(z)G_c(z)\big(s(z) - d(z)\big) + d(z). \tag{5}$$

Therefore, the closed-loop system is stable if $G_c(z)$ and $G(z)$ are stable.

We will assume that the process $G(z)$ is stable. Therefore, stability of the controller $G_c(z)$ is sufficient for closed-loop stability. For the multivariable FMC, the controller transfer function, $G_c(z)$ can be shown to be [7],

$$G_c(z) = \Big[A_P + H_{P+1}z^{-1} + \cdots + H_N z^{-N+P}\Big]^{(-1)}. \tag{6}$$

The stability of the closed-loop system is determined by examining the poles of $G_c(z)$.

III. STABILITY ANALYSIS

From (6), the controller poles are those of the inverse of a matrix of polynomials in z. Finding this inverse and then its associated poles would be no minor task for square systems. For non-square systems it would be even more difficult.

It is much simpler to avoid taking the inverse by looking at the transmission zeros of the matrix. Holt and Morari [10] show that the poles of the system inverse are the transmission zeros of the system. So an unstable transmission zero of the matrix in (6) would become an unstable pole when the inverse is taken and lead to an unstable system.

A general technique for finding the zeros of a system matrix is developed in MacFarlane and Karcanias [11]. This method can be applied to find the transmission zeros of the matrix in (6). The structure of this problem allows some simplification of the method which greatly reduces the calculations involved.

To find the system zeros by this technique requires finding the system pole polynomial first. The system pole polynomial is the least common denominator of all non-zero minors of all orders of the system matrix. Note, however, that the matrix in (6) has no rational terms within it. The terms are polynomials only. Therefore, the least common denominator of the minors is simply one.

The system zeros are the roots of the system zero polynomial. If the system matrix is of order n, then the system zero polynomial is the greatest common divisor of the numerators of all minors of order n, where the minors have been adjusted to all have the pole polynomial as their common denominator. Since for the FMC the pole polynomial is one, the minors do not need adjustment.

To find the minimum P for stability, the roots of these minors of order n must be found. If all the minors have the same root, it will be a transmission zero and if outside the unit circle it will indicate an unstable system. For the special case of a square system, only one minor need be taken. So for this case the controller poles can be found by setting det($G_c(z)$)=0 as noted by Erickson and Otto [7].

IV. EXAMPLE

Stability analysis for the case of a square system was reported earlier (Erickson and Otto [7]). The Wood and Berry [12] methanol/water distillation column was simulated and found to behave just as the stability analysis outlined above indicates it should.

As a more general case, another process model that has found use in the literature was simulated. The multiproduct ethanol/water pilot-plant distillation column (Ogunnaike and co-workers [13]) in Fig. 3 is modeled by equation (7) where time is measured in minutes. The physical meaning of the variables and nominal operating conditions for the column are given in Table 1.

$$\begin{bmatrix} y_1(s) \\ y_2(s) \\ y_3(s) \end{bmatrix} = \begin{bmatrix} \frac{0.66e^{-2.6s}}{6.7s+1} & \frac{-0.61e^{-3.5s}}{8.64s+1} & \frac{-0.0049e^{-s}}{9.06s+1} \\ \frac{1.11e^{-6.5s}}{3.25s+1} & \frac{-2.36e^{-3s}}{5.0s+1} & \frac{-0.012e^{-1.2s}}{7.09s+1} \\ \frac{-34.68e^{-9.2s}}{8.15s+1} & \frac{46.2e^{-9.4s}}{10.9s+1} & \frac{0.87(11.61s+1)e^{-s}}{(3.89s+1)(18.8s+1)} \end{bmatrix} \begin{bmatrix} m_1(s) \\ m_2(s) \\ m_3(s) \end{bmatrix} + \quad (7)$$

$$\begin{bmatrix} \frac{0.14e^{-12s}}{6.2s + 1} & \frac{-0.0011(26.32s + 1)e^{-2.66s}}{(7.85s + 1)(14.63s + 1)} \\ \frac{0.53e^{-10.5s}}{6.9s + 1} & \frac{-0.0032(19.62s + 1)e^{-3.44s}}{(7.29s + 1)(8.94s + 1)} \\ \frac{-11.54e^{-0.6s}}{7.01s + 1} & \frac{0.32e^{-2.6s}}{7.76s + 1} \end{bmatrix} \begin{bmatrix} d_1(s) \\ d_2(s) \end{bmatrix}$$

The model in (7) was adapted to allow a more general test and to simplify the computations involved. A non-square system was desired so y_3, the temperature of the bottoms composition, was removed from the simulation. This modification corresponds to a situation where the temperature measurement is unavailable, not unrealistic for a real process. Given this situation, we need to know what tuning constants will still lead to stable control.

The stability computations outlined above were carried out on an IBM 4341 Group 12 digital computer system in FORTRAN. In order to find the roots of the associated minor polynomials, the IMSL routine ZPOLR [14] was used. For best accuracy within the routine, the FMC impulse response models were kept to 10 points each. In order to do so and yet still sample fast enough for good results, the time constants of the transfer functions in (7) were divided by two. A sampling period of three minutes was used. For the purpose of the analysis and simulation the disturbance models were ignored. The simulated system is then

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{bmatrix} \frac{0.66e^{-2.6s}}{3.35s + 1} & \frac{-0.61e^{-3.5s}}{4.32s + 1} & \frac{-0.0049e^{-s}}{4.53s + 1} \\ \frac{1.11e^{-6.5s}}{1.63s + 1} & \frac{-2.36e^{-3s}}{2.5s + 1} & \frac{-0.012e^{-1.2s}}{3.55s + 1} \end{bmatrix} \begin{bmatrix} m_1(s) \\ m_2(s) \\ m_3(s) \end{bmatrix} \tag{8}$$

The transfer function model (8) was converted to matrices of impulse response models of 10 samples each. The three 2 x 2 minors obtained will be denoted by the columns of the 2 x 3 system used. Thus M_{12} is the minor obtained from columns 1 and 2. The roots of the polynomials obtained from the minors were checked for stability as P_1 and P_2 were decreased from 10.

For $P_1 \geq 2$ and $P_2 \geq 2$ none of the minors have a root outside the unit circle. At $P_1 = 1$ and $P_2 = 2$ minors M_{12} and M_{13} each have a different root outside the unit circle while minor M_{23} is still stable. Thus the system should still be stable. For $P_2 < 2$ the process deadtimes cause many of the elements of A_P to be zero. The controller calculates its actions using $A_P^{(-1)}$ which does not exist for these values of P. Therefore, the minimum control horizons for this case are $P_{1min} = 1$ and $P_{2min} = 2$. Computer simulation of the process verifies these results.

V. SUMMARY

The multivariable Forward Modeling Controller offers a simplified approach to model-based predictive control. Using the stability analysis technique presented in this paper, one can determine the maximum controller performance from the minimum value of the control horizon, P, for each controlled variable. The technique allows stability analysis of square or non-square systems so that stability in the presence of component failure may be analyzed.

ACKNOWLEDGEMENT

Financial support from the Weldon Spring Endowment Fund is gratefully acknowledged.

REFERENCES

[1] Cutler, C. R. and B. L. Ramaker, "Dynamic matrix control -- a computer control algorithm," *Proc. JACC*, San Francisco, paper WP5-B, 1980.

[2] Richalet, J., A. Rault, J. L. Testud and J. Papon, "Model predictive heuristic control: applications to industrial processes," *Automatica*, 14, 413-428, 1978.

[3] Rouhani, R. and R. K. Mehra, "Model algorithmic control (MAC); basic theoretical properties," *Automatica*, 18, 401-414, 1982.

[4] Garcia, C. E. and M. Morari, "Internal model control. 1. A unifying review and some new results," *Ind. Eng. Chem. Process Des. Dev.*, 21, 308-323, 1982.

[5] Garcia, C. E. and M. Morari, "Internal model control. 2. Design procedures for multivariable systems," *Ind. Eng. Chem. Process Des. Dev.*, 24, 472-484, 1985.

[6] Otto, R. E., "Forward modeling controllers: a comprehensive SISO controller," 1986 AICHE Annual Meeting.

[7] Erickson, K. T. and R. E. Otto, "Development of a multivariable forward modeling controller," *Proceedings of the IFAC Workshop on Model Based Process Control*, Atlanta, Georgia, June 1988.

[8] Marchetti, J. L., D. A. Mellichamp, and D. E. Seborg, "Predictive control based on discrete convolution models," *Ind. Eng. Chem. Process Des. Dev.*, 22, 488-495, 1983.

[9] Stoica, P. and T. Söderström, "Uniqueness of prediction error estimates of multivariable moving average models," *Automatica*, 18, 617-620, 1982.

[10] Holt, B. R., and M. Morari, "Design of resilient processing plants--VI. The effect of right-half-plane zeros on dynamic resilience," *Chem. Eng. Sci.*, 40, 59-74, 1985.

[11] MacFarlane, A. G. J. and N. Karcanias, "Poles and zeros of linear multivariable systems: a survey of the algebraic, geometric and complex-variable theory," *Int. J. Control*, 24, 33-74, 1976.

[12] Wood, R. K. and M. W. Berry, "Terminal composition control of a binary distillation column," *Chem. Eng. Sci.*, 28, 1707-1717, 1973.

[13] Ogunnaike, B. A., J. P. Lemaire, M. Morari and W. H. Ray, "Advanced multivariable control of a pilot-plant distillation column," *AICHE Journal*, 29, 632-640, 1983.

[14] *The IMSL Library Reference Manual*, Ninth edition, Vol. 4, IMSL Inc., 1979.

Table 1. Ogunnaike Column Variable Summary

Variable	Description	Operating steady state
y_1	overhead ethanol mole fraction	0.7
y_2	side stream ethanol mole fraction	0.52
y_3	tray #19 temperature	92° C
m_1	reflux flow rate	0.18 gpm
m_2	side stream product flow rate	0.046 gpm
m_3	reboiler steam pressure	20 psig
d_1	feed flow rate	0.8 gpm
d_2	feed temperature	78° C

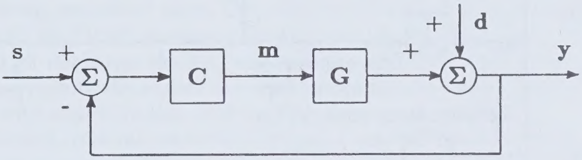


Fig. 1. Conventional control system configuration.

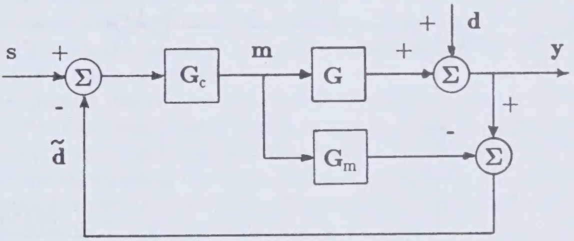


Fig. 2. Internal model control configuration.

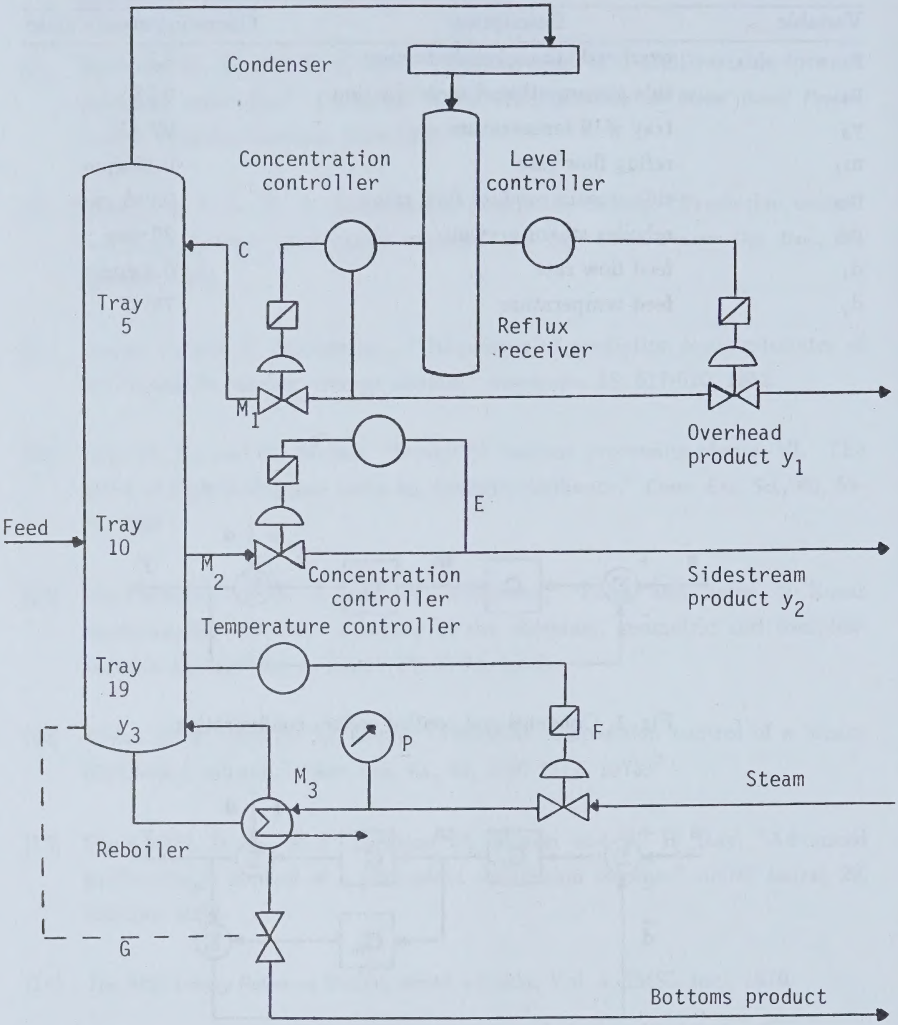

Condenser
Concentration controller
Level controller
C
Tray 5
Reflux receiver
M_1
Overhead product y_1
E
Feed
Tray 10
M_2
Concentration controller
Sidestream product y_2
Temperature controller
Tray 19
y_3
P
F
M_3
Steam
Reboiler
G
Bottoms product

Fig. 3. Diagram of Ogunnaike column.

ADAPTIVE MARKOV DECISION PROBLEMS UNDER CONSTRAINTS

EITAN ALTMAN

Electrical Engineering, Technion – Israel Institute of Technology, Haifa, 32000 Israel

ABSTRACT

We consider the constrained optimization of a finite-state Markov chain. Traditional methods using Dynamic Programming are not applicable in the constrained case and optimal policies are computed through Linear Programs.

We introduce two different methods based on Linear Programming to obtain optimal adaptive policies under some continuity assumptions.

In contrast with the unconstrained problem, applying the control which is optimal for the current parameter estimates ("certainty equivalence" control) may not yield optimal performance. This problem arises since the optimal solution of a Linear Program may exhibit discontinuities as a function of the coefficients.

The theory of sensitivity analysis for Linear Programs is used to modify the "certainty equivalence" approach so that discontinuities do not arise.

INTRODUCTION

The problem of adaptive control of Markov chains has received considerable attention in recent years; see e.g. the survey paper by Kumar [12], Hernandez-Lerma [7] and references therein. In the setup considered there, the transition probabilities of a Markov chain are parameterized, and the "true" parameter value is not known. One then tries to devise a control policy which minimizes the long-run average cost, based on some estimate of the parameters. An asymptotic discounted cost criterion was introduced by Schäl [15]. Adaptive optimal policies with respect to this criterion were investigated in [16,8,7].

We formulate and solve the problem of optimal adaptive control of finite state Markov chains, under average-cost constraints. We assume no prior information about the transition probabilities and their dependence on the control ("non parametric" case). The precise model and basic assumptions are given in Section 1. In Section 2 we describe the classical approach for solving the non-adaptive constrained problem ([6,10]) through Linear Programming. Since the standard approach to adaptive control is based on Dynamical Programming ([12,14,7]), it is not applicable for constrained problems. Shwartz and Makowski introduced a new method [13,17] using the Lagrange approach of Beutler and Ross [4]. This approach is, however, limited to a single constraint.

In Section 3 we present two new methods based on Linear Programs to obtain optimal adaptive policies, both combining probing (which means testing non-optimal actions for the purpose of estimation) with an updating rule of the control. In the first method the probing is applied at some special stopping times. In the second method, at every time instance one randomly chooses between the possibility of probing or the possibility of using the up-to-date rule of the control. The first method has the advantage that it does not need any randomization. The second method can be applied to adaptive problems with other cost criteria, such as the one in [16].

In Section 4 we describe methods for updating the rule of control. If we assume the availability of some "continuous" method to compute policies based on these estimators, then it is shown that the adaptive policy which combines estimation with the substitution of current estimates into the control rule is optimal. However, the "certainty equivalence" approach of using the control which is optimal for the current (estimated) value of the parameters does not, in general, have the desired continuity properties. This can be seen through the computation of controls which is necessarily carried out through a Linear Program whose coefficients are estimated on-line. It is well-known that the optimal solution of

Linear Program may exhibit discontinuities as a function of the coefficients. The theory of sensitivity analysis for Linear Programs [5] is used to modify the previous policies so that discontinuities do not arise. As a result we obtain controls which converge to the optimal policy (i.e. "self tuning" is achieved). In Theorem 3.3 and Theorem 3.8 we prove that this convergence of the controls implies optimality.

1. MODEL AND ASSUMPTIONS.

Consider the discrete time process X_t defined on the finite *state space* $\mathbf{X} = \{1, ..., J\}$. Let A_t be the action chosen at time t, defined on the finite *action space* $\mathbf{A}$. Assume that in any state x all actions in $\mathbf{A}$ are available. Let $H_t := (X_1, A_1, ..., X_t, A_t)$. Define the transition probabilities: $P_{xay} := P(X_{t+1} = y \mid X_t = x; A_t = a) = P(X_{t+1} = y \mid H_{t-1} = h, X_t = x; A_t = a)$.

Define a policy $u = \{u_1, u_2, ...\}$ in the *policy space* U where $u_{t+1}(\cdot \mid H_t, X_{t+1})$ is a conditional probability measure over $\mathbf{A}$. Let $c(x,a)$, $d(x,a) := \{d^k(x,a)\ ,\ k = 1, ..., K\}$ be the cost per state-action pair. Define the cost functions $C(x,u)$ and $D(x,u) := \{D^k(x,u)\ ,\ 1 \leq k \leq K\}$ through

$$C(x,u) = \overline{\lim_{t\to\infty}} \frac{1}{t} E^u \left[\sum_{s=1}^{t} c(X_s, A_s) \mid X_1 = x \right]$$

$D^k(x,u)$ has an analogue definition in term of d^k.

Given the vector $V := \{V_k\ ,\ k = 1, ..., K\}$, define the constrained optimization problem **COP**;

Find $v \in U$ *that minimizes* $C(x,u)$ *subject to*

$$D(x,u) \leq V \tag{1.1}$$

In this paper we assume that the transition probabilities P_{xay} are unknown. We thus solve the adaptive constrained problem **ACOP**, which is to find an optimal policy for COP based on the available information (i.e. using on-line estimation to update and improve the policy). Throughout the paper we impose the following assumptions:

A1: The state space forms a recurrent class under any stationary deterministic policy.

A2: COP is feasible, i.e. there exists some $u \in U$ for which (1.1) holds.

We denote $\delta_a(x)$ the Kronecker delta function, and $|A|$ is the number of elements in a set A. We shall need the following notion of convergence of sets. Given a sequence $\{H_n\}$ of sets in the Euclidean space $\mathbb{R}^l$, define

$\underline{\lim}_{n\to\infty} H_n := \{x \in \mathbb{R}^l : x = \lim_{n\to\infty} x_n$, for some sequence $x_n \in H_n\}$. Similarly,
$\overline{\lim}_{n\to\infty} H_n := \{x \in \mathbb{R}^l$: for some infinite subsequence $x_{n_r} \in H_{n_r}, x = \lim_{r\to\infty} x_{n_r}\}$.

If $\underline{\lim}_{n\to\infty} H_n = \overline{\lim}_{n\to\infty} H_n$ then we denote $\lim_{n\to\infty} H_n = \underline{\lim}_{n\to\infty} H_n = \overline{\lim}_{n\to\infty} H_n$.

2. THE NON-ADAPTIVE CASE

We recall below the standard method for solving COP (see [6,10]). Let z be a matrix taking values in $\mathbb{R}^{|\mathbf{X}|} \times \mathbb{R}^{|\mathbf{A}|}$ and consider the following Linear Program:

*LP*1: Find z that minimizes $\sum_{y\in\mathbf{X}} \sum_{a\in\mathbf{A}} c(y,a) z(y,a)$ subject to:

$$\sum_{y\in\mathbf{X}} \sum_{a\in\mathbf{A}} (\delta_x(y) - P_{yax}) z(y,a) = 0 \qquad 1 \leq x \leq J \tag{2.1a}$$

$$\sum_{y\in\mathbf{X}}\sum_{a\in\mathbf{A}} d^k(y,a)z(y,a) \le V_k \qquad 1\le k\le K \tag{2.1b}$$

$$z(y,a)\ge 0 \quad y\in\mathbf{X}, a\in\mathbf{A} \tag{2.1c}$$

Denote the set of optimal solutions of $LP1$ by B. Note that B is closed and convex. For any z that satisfies (2.1a,2.1c) we define $\gamma(z)$ to be the matrix whose components are given by

$$\gamma_y^a(z) := \frac{z(y,a)}{\sum_{a\in\mathbf{A}} z(y,a)} \tag{2.2}$$

whenever it is well defined. Note that when summing the equality constraints (2.1a) we obtain $\sum_{y\in\mathbf{X}}\sum_{a\in\mathbf{A}} z(y,a)=1$.

Each matrix γ defines a stationary policy, say ν, by assigning the probability of choosing action a when in state y to be $p^\nu_{a|y} = \gamma^a_y$. Denote $\Gamma := \{\gamma(z):\ z\in B\}$. $LP1$ is connected to COP in the following way:

Lemma 2.1: *Under A1, (i) There is at least one z that satisfies constraints (2.1a) and (2.1c). (ii) Any such z has, for each state y, at least one non-zero component $z(y,a)$. (iii) $LP1$ is feasible iff A2 holds. (iv) Consider the stationary policy g, defined by $p^g_{a|y} = \gamma^a_y(z)$. Then $\sum_a z(y,a)$ is the invariant probability measure $\pi^g(y)$ induced by g. Conversely, for each stationary policy g, $z(y,a) := \pi^g(y)\cdot p^g_{a|y}$ satisfies (2.1a, 2.1c). In particular, for each $\beta\in B$, the stationary policy ν defined by $p^\nu_{a|y} = \gamma^a_y(\beta)$ is optimal for COP. Moreover, $\sum_{y\in\mathbf{X}}\sum_{a\in\mathbf{A}} d^k(y,a)\beta(y,a) = D^k(x,\nu)\quad 1\le k\le K$ and $\sum_{y\in\mathbf{X}}\sum_{a\in\mathbf{A}} c(y,a)\beta(y,a) = C(x,\nu)$ for any x.*

Proof: See [10], [6 p. 80]. □

Let $L :=$ the number of linearly independent equations among (2.1a).

Lemma 2.2: *L=J.*

Proof: Suppose $L<J$. Then there exists some z satisfying (2.1a) and (2.1c) with at most L non-zero components. But this contradicts Lemma 2.1 (ii), which concludes the proof. □

In some cases we use the generic notation $g(z)=0$ to denote the equality constraints in LP1, and $f(z)\le 0$ to denote the inequality constraints. The scalar product (e.g. $c\cdot z$ denotes the summation over all common indices.

3 THE OPTIMAL CONTROL.

We propose two adaptive schemes that provide optimal policies for ACOP, both involving estimation of the probabilities which then enables to update and improve the policy.

3.1 Estimation of the transition probabilities. Define:

$$\hat P^t_{xay} := \frac{\sum_{s=2}^t 1\{X_{s-1}=x, A_{s-1}=a, X_s=y\}}{\sum_{s=2}^t 1\{X_{s-1}=x, A_{s-1}=a\}}$$

If the denominator is zero then $\hat P^t$ is chosen arbitrarily, but such that for every x and a, $\hat P^t_{xay}$ is a probability measure. In order that $\lim_{t\to\infty}\hat P^t_{xay} = P_{xay}\quad P^u$ *a.s.* for all states $x,y\in\mathbf{X}$ and $a\in\mathbf{A}$, it is sufficient by SLLN that each state is visited i.o. and, moreover, at each state, each action is used i.o. It turns that under A1, each state x is indeed visited i.o. P^u a.s. under any policy u (see [1]). In either methods described below we shall obtain that $\sum_{s=2}^{\infty} 1\{X_{s-1} = y, A_{s-1} = a\} = \infty \quad P^u$ *a.s.* for all $y \in \mathbf{X},\ a \in \mathbf{A}$, implying consistent estimation.

3.2 The forced choice scheme. To obtain an optimal policy for ACOP, the estimator $\hat{P}^t$ is combined with an updating rule for the control. Clearly, the resulting policy cannot be stationary. Yet we would like to be able to apply the $LP1$ to obtain optimal policies. In [1] Altman and Shwartz generalize the stationary policies as follows. Define

$$f^t(a|y) := \frac{\sum_{s=1}^{t} 1\{X_s = y, A_s = a\}}{\sum_{s=1}^{t} 1\{X_s = y\}}$$

if $\sum_{s=1}^{t}\{X_s = y\} = 0$ set $f^t(a|y) := \frac{1}{|\mathbf{A}|}$. Altman and Shwartz prove the following [1]:

Lemma 3.1: *The costs under a stationary policy ν are also achieved by any other policy u for which $\alpha_y^a := \lim_{t\to\infty} f^t(a|y) = p^{\nu}_{a|y}$ P^u a.s. for all y, a. Consequently, any policy for which such a limit α exists and satisfies $\alpha_y^a = \gamma_y^a$, for some $\gamma \in \Gamma$ is optimal for COP.*

We shall call the policies that are introduced in Lemma 3.1 "Action Time Sharing" (ATS) policies with parameter α By Lemma 3.1 the cost of an ATS policy depends only on the value of α. Therefore, they are not sensitive to non-optimal controls used at some initial finite time interval, or to non-optimal controls provided that the frequency they are used decreases to zero. Note that the costs $C(x, u)$ and $D(x, u)$ also have these properties (due to the time-average in their definition). Thus it is natural to apply methods using ATS policies for ACOP since it enables adequate learning of the system by probing without affecting the cost. This approach is thus applied in Algorithm 3.2 below.

Suppose we are given a sequence $\Phi := \{\Phi^r, r = 1, 2, ...\}$ (called a "control law") of approximations of an optimal $\gamma \in \Gamma$ (specific control laws are introduced in Section 4), where each Φ^r depends on the estimates of $\{P_{xay}\}$. Let $d(z, z') := \max\{|z(y, a) - z'(y, a)| : y \in \mathbf{X}, a \in \mathbf{A})\}$. Fix a sequence $\epsilon_r \downarrow 0$ and define the sequences of stopping times $\{T_i\}_{i=1}^{\infty}$ and $\{S_i\}_{i=1}^{\infty}$ as follows: Set $T_1 := 0$. Define $S_r := \inf\{t > T_r$; each state has been visited at least $|\mathrm{A}|$ times after $T_r\}$. $T_r := \min\{t > S_{r-1} : d[f^t(\cdot|\cdot), \gamma(r-1)] < \epsilon_r\}$, $r \geq 2$ where $\gamma(r)$ is obtained at time S_r using the control law Φ^r. The meaning of these stopping times is as follows: The period between T_r and S_r is devoted for estimation and control. At S_r the policy is updated. The period between S_r and T_{r+1} is devoted for control.

In the following Algorithm 3.2 the r^{th} estimate of $\{P_{xay}\}$ is substituted in Φ^r to yield a series $\gamma(r)$ of approximations of some $\gamma \in \Gamma$. The performance of the Algorithm is obtained in Theorem 3.3 below.

Algorithm 3.2: (1) Set $r = 1$ and choose $\gamma(0)$ arbitrarily. (2) Probing (estimation) is obtained by choosing action a_i at the ith visit to state y after T_r (where the actions in $\mathbf{A}$ have been ordered as $a_1, ..., a_{|\mathbf{A}|}$). (3) Except at "probing times" choose an action as follows: if $X_t = y$ then $a' \in \operatorname{argmin}\{a \in \mathbf{A} : f^t(a|y) - \gamma_y^a(r-1)\}$. Continue until S_r. (4) Calculate P^{S_r} to obtain $\gamma(r) = \Phi^r(\hat{P}^{S_r})$. If $X_t = y$ then choose $a' \in \operatorname{argmin}\{a \in \mathbf{A} : f^t(a|y) - \gamma_y^a(r)\}$. Continue until T_r, increase r by 1 and repeat from step 2.

Theorem 3.3: *Under A1, (i) $T_r < \infty$ and $S_r < \infty$ for all r (w. p. 1), and (ii) $\hat{P}^{T_r} \to P$ w.p.1. Assume moreover A2 and that under the control law Φ, (ii) implies that $\gamma(r)$ converges to some $\gamma \in \Gamma$. Then Algorithm 3.2 yields an optimal policy for ACOP.*

Proof: (i) can easily be proved by induction. (ii) is then obtained by SLLN since at any state y, all actions are used infinitely often (w.p.1) due to the first claim. By (ii), the definition of T_r and the hypothesis, under Algorithm 3.2 $\lim_{r\to\infty} f^{T_r}(a|y) = \gamma_y^a$ for all $a \in \mathbf{A},\ y \in \mathbf{X}$. It can be shown that in fact $\lim_{t\to\infty} f^t(a|y) = \gamma_y^a$ for all $a \in \mathbf{A},\ y \in \mathbf{X}$. Hence the adaptive policy is in fact an ATS policy with parameter γ. by Lemma 3.1 and from $LP1$, this policy is optimal. The details are omitted. (A detailed proof of a similar Theorem can be found in [3]). □

Note that Algorithm 3.2 does not require any randomization.

3.3 The randomization scheme. In order to develop adaptive policies that are also suitable for for adaptive problems with asymptotic discounted cost (see e.g. [16]) ATS policies are not suitable since discounted cost criteria are sensitive to probing, e.g. if we use i.o. non-optimal actions then the cost achieved will not be optimal. We therefore propose a new class of policies that are applied in Algorithm 3.4 below. They enable to solve ACOP and can also be used in problems with asymptotic costs (such as Schäl's cost criteria). For each stationary policy g we denote by $\hat{\mathbf{G}}$:= the set of *asymptotically stationary* policies $\{u\}$ which satisfy $\lim_{t\to\infty} P^u(A_t = a|X_t = y, X_1 = x) = p^g_{a|y}$ for any initial state x and all a and y. We shall need in this Section to strengthen A1; assume

A1': The state space forms an aperiodic recurrent class under any stationary deterministic policy.

As in Section 3.2, suppose we are given a sequence $\Phi = \{\Phi^r\ ,\ r = 1, 2, ...\}$ of approximations of some $\gamma \in \Gamma$. Fix a sequence $\epsilon_r \downarrow 0$ such that $\sum_1^\infty \epsilon_r = \infty$, and an increasing sequence of times T_r with $T_1 = 0$. T_r are the times at which we update the estimate. One can choose $T_r = q \cdot r$ where q is some positive integer that is proportional to the computation time of Φ^r. In the following Algorithm we construct a policy u which is shown in Theorem 3.8 to be optimal for ACOP.

Algorithm 3.4: If $X_t = x$ and x has been visited n times, and $T_r \le t < T_{r+1}$ then u_t is defined through:

$$u_t(\cdot|H_{t-1}, X_t) = \frac{\epsilon_n}{|\mathbf{A}|} + (1 - \epsilon_n) \cdot \Phi^r(\hat{P}^{T_r}) \tag{3.1}$$

We present some definitions and Lemmas needed for proving the optimality of Algorithm 3.4.

Given a policy u, define $\eta^u(y, a) := \lim_{t\to\infty} P^u(X_t = y, A_t = a)$ whenever this limit exists. Note that under A1' for any stationary policy g, η^g exists and is equal to $\eta^g(y, a) = \pi^g(y) \cdot p^g_{a|y}$.

Lemma 3.5: *Under A1', given any stationary policy g, if a policy u satisfies $u \in \hat{\mathbf{G}}$ then for any initial state x, $\lim_{t\to\infty} P^u(X_t = \cdot|X_1 = x) = \pi^g$ $\ P^u$ a.s., and η^u exists and is equal to η^g.*

To prove Lemma 3.5 we need another Lemma. For any real valued vector $z \in \mathbb{R}^{|\mathbf{X}|}$ let $||z||_1 := \sum_{y\in\mathbf{X}} |z(y)|$. Let $\zeta(\cdot)$ be any probability measure on $\mathbf{X}$.

Lemma 3.6: *Assume that for some stationary policy g, $P^g_{xy} > \epsilon > 0$ for all $x, y \in \mathbf{X}$. Then $||\zeta \cdot P^g - \pi^g||_1 \le (1 - \epsilon)||\zeta - \pi^g||_1$.*

Proof of Lemma 3.6: Note that $||\zeta \cdot P^g - \pi^g||_1 = \sum_y |\sum_x(\zeta(x) - \pi^g(x))P_{xy}|$ and $||\zeta - \pi^g||_1 \sum_x \sum_y |\zeta(x) - \pi^g(x)|P_{xy}$. Let $[\zeta(x) - \pi^g(x)]^- := \max\{0, -[\zeta(x) - \pi^g(x)]\}$. We then obtain

$$||\zeta - \pi^g||_1 - ||\zeta \cdot P^g - \pi^g||_1 = \sum_y \sum_x 2[\zeta(x) - \pi^g(x)]^- \cdot P^g_{xy} \ge 2\epsilon \sum_x [\zeta(x) - \pi^g(x)]^-$$

Since $\sum_x[\zeta(x) - \pi^g(x)] = 0$ it follows that $||\zeta - \pi^g||_1 - ||\zeta \cdot P^g - \pi^g||_1 = \epsilon||\zeta - \pi^g||_1$ and the Lemma easily follows. □

Proof of Lemma 3.5: Under g, $\{X_t\}$ is a regular Markov chain; hence for some integer L, $[P^g]^L$ has all components nonzero (see e.g. [11 p. 69]). Let $P(t)$ be the matrix whose components are given by: $P_{zy}(t) := P^u(X_t = y | X_{t-1} = z, X_1 = x) = \sum_a P_{xay} \cdot P^u(A_t = a | X_t = z, X_1 = x)$. Let ρ_j be a row vector with 1 in the j-th component and 0 otherwise. Denote $Q_x^t := P^u(X_t = \cdot \, | X_1 = x)$. We observe that $Q_x^t = Q_x^{t-1} \cdot P^u(t)$ and therefore $Q_x^t = \rho_x \cdot \Pi_{s=0}^t P(s)$.

There exists some t_0 such that for all $t > t_0$ $R(t) := \Pi_{s=t+1}^{t+L} P(t)$ has all components nonzero, and, by continuity of $\{P(S)\}_{s=t+1}^{t+L} \to \Pi_{s=t+1}^{t+L} P(S)$, all components of $R(t)$ are greater than some positive ϵ'. Denote $\theta := 1 - \epsilon'$. Denote π^t the stationary probability of a Markov chain whose transition probabilities are given by $R(t)$. (π^t exists and is unique for all $t > t_0$). Since $P(t)$ converges to P^g, it follows that $R(t) \to [P^g]^L$ and hence (see e.g. [9]) $\lim_{t \to \infty} \pi^t = \pi^g$ since π^g is the invariant distribution of $[P^g]^L$. Pick some $\epsilon > 0$. It then follows that there exists some $t_1 > t_0$ such that $\|\pi^t - \pi^g\|_1 \leq \epsilon$ for all $t > t_1$. For $t \geq t_1$ it can be shown using Lemma 3.6 that: $\|Q_x^{t+nL} - \pi^g\|_1 \leq \theta \|Q_x^{t+(n-1)L} - \pi^g\|_1 + \epsilon$. By iteration we then obtain:

$$\|Q_x^{t+nL} - \pi^g\|_1 \leq \theta^n \|Q_x^t - \pi^g\|_1 + \frac{\epsilon}{1-\theta} \leq \theta^n + \frac{\epsilon}{1-\theta}$$

Hence $\lim_{t \to \infty, n \to \infty} \|Q_x^{t+nL} - \pi^g\|_1 = 0$ from which we obtain $\lim_{t \to \infty} Q_x^t = \pi^g$. This proves the first claim. Since $P^u(X_t = y, A_t = a | X_1 = x) = P^u(X_t = y | X_1 = x) \cdot P^u(A_t = a | X_t = y, X_1 = x)$, η^u exists and is equal to η^g. □

Lemma 3.7: *Given a stationary policy g, if a policy u satisfies $u \in \hat{\mathbf{G}}$ then $C(x,u) = C(x,g)$.*

Proof: For any policy w, $C(x,w)$ can be represented through

$$C(x,w) = \overline{\lim_{t \to \infty}} \sum_{y,a} c(y,a) \left[\frac{1}{t} \sum_{s=1}^{t} P^w(X_s = y, A_s = a \mid X_1 = x) \right]$$

Pick some $u \in \hat{\mathbf{G}}$. From Lemma 3.5 we have $\eta^u = \eta^g$ and hence

$$\lim_{t \to \infty} \frac{1}{t} \sum_{s=1}^{t} P^u(X_s = y, A_s = a \mid X_1 = x) = \eta^g(y,a)$$

It then follows that $C(x,u) = \eta^u \cdot c = \eta^g \cdot c = C(x,g)$ which establishes the Lemma.

Theorem 3.8: *Assume A1' and A2, and that $\hat{P}^{T_r} \to P$ w.p.1. implies that $\Phi^r(P^{T_r})$ converges to $\gamma \in \Gamma$. Let g be the stationary policy that corresponds to γ (through Lemma 2.1 (iv)). Then the policy u obtained through Algorithm 3.4 satisfies $u \in \hat{\mathbf{G}}$ and is optimal for ACOP.*

Proof: Pick some state y and an action a. Recall that y is visited i.o. P^u a.s. (Section 3.1). Define $\tau_0 := 0$ and $\tau_{n+1} := \inf\{t > \tau_n : X_t = y\}$. According to (3.1) and the construction of $\{\epsilon_n\}$ we have $\sum_{n=1}^{\infty} P(A_{\tau_n} = a) = \infty$. Hence by Borel Cantelli's Lemma, at state y action a is used i.o. Since this holds for any y and a it follows that $\hat{P}^{T_r} \to P$ w.p.1. (see Section 3.1). Hence by hypothesis $\Phi^r(P^{T_r})$ converges to γ, which implies by (3.1) that $u_t(\cdot | H_{t-1}, X_t = z) \to p^{\gamma}_{\cdot|z}$. It can then be shown that

$$lim_{t \to \infty} P^u(\cdot | X_t = z, X_1 = x) = p^{\gamma}_{\cdot|z}$$

Hence $u \in \hat{\mathbf{G}}$ is an asymptotically stationary policy and by Lemma 3.7 it is optimal for *ACOP*.

4. THE CONTROL LAWS

In order to construct an optimal policy, we needed (Theorems 3.3 or 3.8) some control law Φ that consists of a converging sequence of approximations Φ^n for γ. This is done below by computing a sequence $\beta(n)$ of approximations of some $\beta \in B$ from given estimates of the unknown coefficients of LP1. Φ^n is then computed by $Phi^n := \gamma(\beta(n))$ through (2.2).

Below we formulate the obvious "certainty equivalence" approach of substituting the estimates in $LP1$, and describe the resulting difficulties. A modified control law is then introduced, and in Theorem 4.5 we prove the convergence of $\beta(n)$ to β under the additional assumptions A2′ , A4 below. Theorem 3.3 (3.8) then imply that the adaptive policy obtained by Algorithm 3.2 (3.4 respectively) is optimal for ACOP.

A2′: COP is feasible when replacing the inequalities in (1.1) by strict inequalities, i.e. $D_x(u) < V$.
A4: LP1 has only one solution, i.e. $B = \{\beta\}$.

Note that A4 or A2′ imply assumption A2.

4.1: Certainty Equivalence approach The "certainty equivalence" Φ_{ce} control for this problem is obtained by substituting the estimates generated by Algorithm 3.2 (or 3.4) into LP1, and thus computing the control. For both methods, the Linear Program takes the form

LP1n : Find z^n that minimizes $c \cdot z$ subject to:

$$g^n(z) = 0, \qquad z \geq 0, \qquad d^k \cdot z \leq V^k \qquad 1 \leq k \leq K$$

Note that only the equality constraints differ from those in LP1. Choose $\beta(0)$ arbitrarily but such that $\beta_i \geq 0 \quad i \leq i \leq l$ and $\sum_{i=1}^{l} \beta_i = 1$. Let $B(n)$ be the set of solutions of $LP1n$. If $B(n)$ is empty then set $\beta(n) := \beta(n-1)$. Otherwise, pick any element of $B(n)$ and denote it by $\beta(n)$.

We use the well known theory of sensitivity analysis in Linear Programs. We begin by presenting a lemma obtained by Dantzig, Folkman and Shapiro [5], which gives conditions for the convergence of $\beta(n)$ to β - the solution of LP1.

Let f, f^n, g, g^n be affine functions from $\mathbb{R}^l$ to $\mathbb{R}^m$ and $\mathbb{R}^{m'}$ respectively; throughout this section we assume that $f^n \to f$, and $g^n \to g$ pointwise. Given a Linear Program (e.g. LP1) with l decision variables, $f = (f_1, f_2, ..., f_m)$ (and f^n) represent the m inequality constraints and g (g^n) represents the m' equality constraints.

Given any function σ from $\mathbb{R}^l$ to $\mathbb{R}$ and a set $H \in R^l$, define $M(\sigma|H)$ to be the subset of H where σ achieves its minimum, i.e. $M(\sigma|H) := \{x \in H : \sigma(x) = \inf\{\sigma(y)|y \in H\}\}$. Denote $H(f,g) := \{x \in \mathbb{R}^l \mid f(x) \leq 0, \text{ and } g(x) = 0\}$; this is the feasible set of the Linear Program associated with f, g.

Lemma 4.2: *Assume that $lim_{n\to\infty} H(f^n, g^n) = H$ for some set H. Let σ and $\{\sigma_n\}$ be linear functions such that $\sigma_n \to \sigma$ pointwise. Then*

$$\overline{\lim_{n\to\infty}}\, M(\sigma_n|H(f^n, g^n)) \subset M(\sigma|H) \tag{4.1}$$

Moreover, the left side of (4.1) is nonempty.

Proof: The first claim follows immediately from [5] Theorem I.2.2. The second claim holds since the feasible set of LP1 is contained in a simplex so that $M(\sigma_n|H(f^n,g^n))$ are uniformly bounded. $\square$

Thus the convergence of the optimal set of a Linear Program depends on the convergence of the feasible set. Next we quote another result from [5] that gives conditions for this convergence. Denote $I := \{i \mid 1 \leq i \leq m, f_i(x) = 0 \text{ for all } x \in H(f,g)\}$, $f_I := \{f_i\ ,\ i \in I\}$, rank$(f_I, g)$:= the rank of the matrix whose $|I| + m'$ rows are the coefficiences of the linear functions $f_i, i \in I$, and $g_j, 1 \leq j \leq m'$.

Assume **A5:** $\overline{\lim}_{n\to\infty} \operatorname{rank}(f_I^n, g^n) \leq \operatorname{rank}(f_I, g)$.

Lemma 4.3: ([5] Cor. II.3.4) *Under A2,A5 either* $\lim_{n\to\infty} H(f^n,g^n) = H(f,g)$ *or* $H(f^n,g^n)$ *is empty for infinitely many n.*

Using Lemma 4.2 and Lemma 4.3, the following Theorem gives conditions for $\overline{\lim}_{n\to\infty}\{\beta(n)\} \subset B$, and for the convergence of $\beta(n)$ to β, given that $B = \{\beta\}$. Using Theorems 3.3 or 3.8, the last convergence implies optimality of the self tuning approach. Define f, g (f^n, g^n) as the inequality and equality constraints in LP1 ($LP1n$ respectively).

Theorem 4.4: *Under A2,A5 if* $H(f^n,g^n)$ *is non empty i.o. then* $\overline{\lim}_{n\to\infty}\{\beta(n)\} \subset B$ *and the limit set is non empty.*
Assume moreover that A4 holds. Then $\lim_{n\to\infty} \beta(n) = \beta$.

Proof: If $H(f^n,g^n)$ is i.o. non empty, we have by Lemma 4.3, along the subsequence where $H(f^n,g^n)$ is not empty, $H(f^n,g^n) \to H(f,g)$. Hence by Lemma 4.2 $\overline{\lim}_{n\to\infty} B(n) \neq \phi$. Recall that $\beta(n)$ is updated only when $B(n)$ is not empty. Hence Lemma 4.2 implies $\overline{\lim}_{n\to\infty}\{\beta(n)\} \subset B$, which establishes the first claim. Since by A4 $B = \{\beta\}$, it follows that $\overline{\lim}_{n\to\infty}\{\beta(n)\} \subset \{\beta\}$ and since the set on the left is non empty, $\overline{\lim}_{n\to\infty}\{\beta(n)\} = \{\beta\}$. But this means that all possible subsequences of $\{\beta(n), n = 1, 2, ...\}$ converge to β, so that $\lim_{n\to\infty} \beta(n) = \beta$. $\square$

From Theorem 4.4, two problems are seen to arise in attempting to apply the self tuning approach: (i) As the coefficients in LP1 are not known we may not be able to verify A5. (ii) At any finite time, we have no knowledge whether H^n is going to be non-empty i.o. and thus, even when A5 holds we cannot be sure that $\lim_{n\to\infty} H(f^n,g^n) = H(f,g)$.

An example where the self tuning approach can be used successfully, is the unconstrained problem (i.e. ACOP with K=0). In that case $H(f^n,g^n)$ is never empty (Lemma 2.1 (i)). Moreover, A5 holds since the only inequality constraints in $LP1$ and $LP1n$ are $z \geq 0$. As $LP1$ clearly remains feasible when imposing strict inequality in that constraint it follows that $I = \phi$ (see Lemma 4.6). Since g is full rank (see Lemma 2.2) A5 follows. Hence it follows from Theorem 4.4 that under A4 $\beta(n) \to \beta$, and the certainty equivalence approach yields optimal adaptive control.

Next, we suggest an improved control law based on relaxing the constraints. Under this control, problem (ii) does not arise, while conditions for A5 are given in Lemma 4.6.

4.2 Relaxing the constraints.

The second control law Φ_{rc} is obtained by first solving the following Linear Program:

LP2n′ : Find z and η that minimize η subject to:

$$g^n(z) = 0 \tag{4.2a}$$

$$z \geq 0 \quad , \quad \eta \geq 0 \tag{4.2b}$$

$$d^j \cdot z \leq V^j + \eta \qquad (4.2c)$$

Denote the optimal η by η^n. Then solve the following Linear Program:

LP2n : Find z that minimizes $c \cdot z$ subject to:

$$g^n(z) = 0 \qquad (4.3a)$$

$$z \geq 0 \qquad (4.3b)$$

$$d^j \cdot z \leq V^j + \eta^n \qquad (4.3c)$$

Denote any solution of $LP2n$ by $\beta(n)$. Let H_n denote the set of z satisfying (4.3a-4.3c), and H'_n the set of (z, η) satisfying (4.2a-4.2c). Let LP2′ (LP2) denote the Linear Program obtained by replacing g^n by g (and replacing η_n by 0) in $LP2n'$ ($LP2n$ respectively). Define f (f^n) as the respective inequality constraints in $LP2$ ($LP2n$ respectively). Assume $g^n \to g$. Note that g and f are exactly the functions defining $LP1$.

Theorem 4.5: *Assume A1,A2,A5. Then* H_n and H'_n *are non empty for each n. Moreover,* $\overline{\lim}_{n\to\infty}\{\beta(n)\} \subset B$, *and under A4* $\beta(n) \to \beta$ *(the solution of LP1).*

Proof: First claim: when omitting the constraints (4.2c) and setting $\eta = 0$, any stationary policy yields a feasible solution for $LP2n'$ (by Lemma 2.1 (iv)). Let $\{b, 0\}$ be one such solution. Since the state and action spaces are finite, there exists some positive constant L such that $D_x^k(u) \leq L$ for all policies u and all k. Thus by choosing $\eta' = L + \max_k\{| V^k |\}$ we have $(b, \eta') \in H'_n$, so H'_n is nonempty. Now set $S_n = \{z : (z, \eta^n) \in H'_n\}$. Then $S_n \times \{\eta^n\}$ is a subset of the optimal solutions for $LP2n'$, and it clearly is non-empty. But $H_n = S_n$ which establishes the first claim.

To prove the second claim, note that $\eta^n \to 0$. This can easily be proved by the previous argument and Lemma 4.2. It then follows that f^n converges to f. Since H_n is non-empty, it converges by Lemma 4.3. Finally, it follows from Lemma 4.2 that $\overline{\lim}_{n\to\infty}\{\beta(n)\} \subset B$, and under A4 $\beta(n) \to \beta$ (as in the proof of Theorem 4.4). ☐

Lemma 4.6: A2′ *is equivalent to* $I = \phi$, *and implies A5.*

Proof: Since g is full rank (see Conclusion in Section 2) $I = \phi$ implies A5. Since the feasible set of LP1 is convex, $I = \phi$ is equivalent to the requirement that there should exist some z that satisfies the inequality inequality constraints in LP1 with the inequalities replaced by strict inequalities. From Lemma 2.1 (iv) it follows that $I = \phi$ implies A2′. On the other hand, we conclude from Lemma 2.1 (iv) that A2′ implies the existence of some stationary policy v whose corresponding z satisfies

$$d^k \cdot z < V_k \qquad 1 \leq k \leq K \qquad (4.4)$$

It remains to show that there exists some z' satisfying both (4.4) and

$$z' > 0, \qquad g(z') = 0 \qquad (4.5)$$

which corresponds to some other stationary policy.

By construction, z satisfies (4.4) and $g(z) = 0$ (z is obtained by $z(y,a) = \pi^v(y) \cdot p^v_{a|y}$). Pick any other stationary policy w that satisfies $p^w_{a|y} > 0, \quad a \in \mathbf{A}, y \in \mathbf{X}$. Let $z''(y,a) = \pi^w(y) \cdot p^w_{a|y}$. By assumption A1 and Lemma 3.1, z'' satisfies $z'' > 0$ and $g(z'') = 0$. Let $z' := (1-\delta)z + \delta z''$, where δ is some positive constant. By choosing δ small enough, z' satisfies (4.4)-(4.5). ☐

Conclusion: We presented two control laws that produce a series $\beta(n)$ of approximations of optimal solutions of LP1. The required control law is obtained by computing $\gamma(\beta(n))$ through (2.2). Under assumptions A1, A2$'$ and A4 we obtained a sequence $\beta(n)$ that converges to the required β which implies $\Phi^n \to \gamma$. Under these assumptions one can compute optimal adaptive policies through Algorithm 3.2 (see Theorem 3.3) or, under the additional assumption A1', through Algorithm 3.4 (see Theorem 3.8).

REFERENCES

[1] Altman E. and A. Shwartz, "Non-stationary policies for controlled Markov Chains", June 1987, submitted.

[2] Altman E. and A. Shwartz, "Markov decision problems and state-action frequencies", EE. PUB. No. 693, Technion, November 1988, submitted.

[3] Altman E. and A. Shwartz, "Adaptive control of constrained Markov chains", EE. PUB. No. 713, Technion, March 1989, submitted.

[4] Beutler F. J. and K. W. Ross, "Optimal policies for controlled Markov chains with a constraint", *Math. Anal. Appl.* Vol. 112, pp. 236-252, 1985.

[5] Dantzig G. B., J. Folkman and N. Shapiro, "On the continuity of the minimum set of a continuous function", *J. Math. Anal. and Applications,* Vol. 17, pp. 519-548, 1967.

[6] Derman C., *Finite State Markovian Decision Processes,* Academic Press, 1970.

[7] Hernandez-Lerma O., *Adaptive Control of Markov Processes*, Preprint, 1987.

[8] Hernandez-Lerma O. and S. I. Marcus, "Adaptive Control of Discounted Markov Decision Chains", *J. Opt. Theory Appl.* Vol. 46 No. 2, pp. 27-235, June 1985.

[9] Hordijk A., *Dynamic Programming and Markov potential theory* , Mathematical Center Tracts no. 51, Amsterdam, 1974.

[10] Hordijk A. and L. C. M. Kallenberg, "Constrained undiscounted stochastic dynamic programming", *Mathematics of Operations Research,* Vol. 9, No. 2, May 1984.

[11] Kemeny J. G. and J. L. Snell, *Finite Markov Chains* , D. Van Nostrand Company, 1960.

[12] Kumar P. R., "A survey of some results in stochastic adaptive control", *SIAM J. Control Opt.*, Vol. 23 No. 3, pp. 329-380, May 1985.

[13] Makowski A. M. and A. Shwartz, "Implementation issues for Markov decision processes", *Proc. of a workshop on Stochastic Differential Systems, Stochastic Control Theory and Applications,* Inst. Math. Appl., Univ. of Minnesota, Springer-Verlag Lecture Notes, Control and Inf. Sci., Edited by W. Fleming and P.-L. Lions, 1986.

[14] Militio R. A. and J. B. Cruz, Jr., "An optimization-oriented approach to the adaptive control of Markov chains", *IEEE Trans. Automatic Control,* Vol. 32 No. 9, pp. 754-762, Sept. 1987.

[15] Schäl M., "Conditions for optimality in dynamic programming and for the limit of n-stage optimal policies to be optimal", *Z. Wahrscheinlichkeitstheorie und verw. Geb.* Vol. 32, pp. 179-196, 1975.

[16] Schäl M., "Estimation and control in discounted dynamic programming," *Stochastics* **20**, pp. 51-71, 1987.

[17] Shwartz A. and A. M. Makowski, "An Optimal Adaptive Scheme for Two Competing Queues with Constraints", *Analysis and Opt. of Systems,* edited by A. Bensoussan and J. L. Lions, Springer Verlag lecture notes in Control and Info. Scie. No. 83, 1986.

ON THE MAXIMUM QUEUE LENGTH WITH APPLICATIONS TO DATA STRUCTURES ANALYSIS

WOJCIECH SZPANKOWSKI†

Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA

Abstract. *A dynamic data structure called queue is analyzed in this paper from the viewpoint of its maximum size. By dynamic queue we understand any data structure that is built during a sequence of insertions and deletions. The maximum size of such a structure is a fundamental quantity and is directly related to many problems of resource allocations. We assume that each element of the structure (we call it further a customer) stays for a random time in the system and then leaves it. Furthermore, the interarrival time of customers is a generally distributed random variable. Adopting queueing theory language, we say the data structure is GI/G/c queueing systems, where c (may be infinite) is the maximum number of items that can simultaneously leave the system (number of servers). We shall show that for stable stationary queue the maximum queue length observed by the n-th arriving customer grows asymptotically in probability as* $\log_\alpha n$*, where* α *is a system constant.*

1. MOTIVATIONS AND INTRODUCTION

Queues and other data structures [AHU] behaviors are natural models for many dynamic phenomena. They correspond to important processes in the areas of algorithms, operating systems, distributed systems, computer networks, etc. Queues are understood here as a dynamic data structure with random insertions and deletions, that is, an item may arrive at any random moment of time and may stay in the structure for a random time (service time) until it is deleted (served). We do not impose any special restrictions on the distributions of the interarrival times of items and service times. In queueing theory terminology, we consider GI|G|c systems [ASM, KLE] where *c* (may be infinite) stands for the number of servers, and GI/G means respectively that the interarrival and service times are generally distributed.

Dynamics of queues can be studied through the transient analysis, however, this seems to be hopeless in our general setting [cf. ASM]. Nevertheless, some important information about dynamics of the structure can be obtained by analyzing the maximum size of the queue over a period of time. Such information, without any doubt, has obvious significance to issues of resource allocation (e.g., the design of a buffer size in a distributed system). The maximum queue length was extensively studied in the seventieth by queueing theoreticians. Heyde [HEY] was the first who predicted the asymptotic growth of maximum queue length in the G|M|1 system. His result was extended by Cohen in [COH] to M|G|1 systems, and finally Iglehart [IGL] completed these studies by providing the growth of rate for GI|G|1 queue. Unfortunately, all the works but Heyde are rather limited to queueing theory and the methodology is too complicated, with no clear extensions to other dynamic

---

† This research was supported in part by NSF under grants NCR-8702115 and CCR-8900305.

structures. In this paper, we provide a new simple methodology to study the maximum queue length in GI|G|1 and PH|PH|c [TAK] queueing systems. The bad news from this analysis (cf. [HEY, COH, IGL]) is that the maximum queue length observed by the n-th arriving customer grows asymptotically *in probability* like $\log_\alpha n$, where α is a system parameter depending upon the interarrival and service distribution functions. In addition, the proposed methods is robust in that it applies to several different models, including discrete-time priority queue [AHU], hashing with lazy deletion [MSW, WV], geometric adjacency problem that arose in the analysis of VLSI [SW], performance evaluation of digital trees [SZ1], graph optimization problem [SZ2], etc.

This paper is organized as follows. In the next section, after presenting some preliminary results form queueing theory, we state our main results. In Section 3 we apply these results to study hashing with lazy deletion.

2. MAIN RESULTS

In this section we establish our main results. We begin with some preliminary definitions from queueing theory, then we formulate our proposition, and finally we discuss some consequences of our main results. All proofs is presented in the last two subsections, each one dealing with upper and lower bounds on the maximum queue length.

2.1 Preliminary Results

We analyze a single server queueing system with arbitrary interarrival times and service times, that is, GI|G|1 model. Let $A(t)$ and $B(t)$ respectively, represent distribution functions of the interarrival times and service times. We denote by $A^*(s)$ and $B^*(s)$ the corresponding Laplace-Stieltjes transforms of $A(t)$ and $B(t)$. The interarrival times, as well as the service times, are mutually independent, that is, both processes form a renewal process. Two quantities are of particular interest, namely the queue length Q_k and waiting time W_k at the moment of the k-th arrival of a customer. Our purpose is to estimate

$$\overline{Q}_n = \max_{1 \le k \le n} Q_k \tag{2.1a}$$

$$\overline{W}_n = \max_{1 \le k \le n} W_k \tag{2.1b}$$

as n tends to infinity. More precisely, we establish asymptotic growth of the r-th moments $E\overline{Q}_n^r$ and $E\overline{W}_n^r$ of the maximum queue length and waiting time, as well as convergence of these quantities in probability.

It turns out that the asymptotics of $\overline{Q}_n$ and $\overline{W}_n$ depend on the tail of the stationary distributions of the queue length and the waiting time. In the further part of this paper, we assume that the system is *stable*, that is, $\rho = \lambda/\mu < 1$ with λ and μ being intensities of the arrival and service processes respectively. Under this condition, a stationary distribution exists [ASM, KLE], and we deal further only with the *stationary processes* Q_k and W_k. Then, we denote by $Q(m) = Pr\{Q_k \le m\}$ and $W(x) = Pr\{W_k < x\}$, respectively, the distribution functions of the queue length Q_k and the waiting time W_k. The following well known result of Feller [FEL] describes the tail distributions of these two processes. Let θ be a unique positive solution, if it exists, of the following complex equation

$$A^*(\theta)B^*(-\theta) = 1 \tag{2.2}$$

Then, Feller proves that for $x \to \infty$

$$1 - W(x) = c_1 e^{-x\theta}(1 + o(1)) \tag{2.3}$$

or in another notation $1 - W(x) \sim c_1 e^{-x\theta}$ for $x \to \infty$. As a simple consequence of (2.3), we obtain the tail distribution of the queue length. Indeed, define ω as

$$\omega = A^*(\theta) < 1 \tag{2.4}$$

then for $m \to \infty$

$$1 - Q(m) = c_2 \omega^m (1 + o(1)) \tag{2.5}$$

where c_1 and c_2 are constants.

The above results have been recently extended to some c-servers queueing systems, namely PH|PH|c where PH stands for phase distribution (see [TAK, ASM, KLE]). Takahashi proved that for FIFO PH|PH|c systems, the following holds

$$1 - W(x) \sim c_3 e^{-x\theta c} \tag{2.6a}$$

$$1 - Q(m) \sim c_4 \omega^m \tag{2.6b}$$

where θ and ω are defined respectively as

$$A^*(c\theta) B^*(-\theta) = 1 \tag{2.7a}$$

$$\omega = A^*(c\theta) = 1/B^*(-\theta) \tag{2.7b}$$

Using the above asymptotics we prove our main results.

PROPOSITION. For GI|GI|1 and PH|PH|c with c finite the following holds *provided* equations (2.2) and (2.7) possess solutions.

(i) The r-th moments $E\overline{Q}_n^r$ and $E\overline{W}_n^r$ of the maximum queue length and waiting time become

$$E\overline{Q}_n^r = \log_\omega^r n^{-1}(1 + o(1)) \tag{2.8a}$$

$$E\overline{W}_n^r = \log^r n^{1/c\theta}(1 + o(1)) \tag{2.8b}$$

for large n.

(ii) The following convergences hold

$$\lim_{n \to \infty} \frac{\overline{Q}_n}{\log_\omega n^{-1}} = \lim_{n \to \infty} \frac{\overline{W}_n}{\log n^{1/\theta}} = 1 \tag{2.9}$$

in probability sense.

■

We prove the Proposition in the next two subsections, however we first note that Proposition (ii) is a simple consequence of Proposition (i) and Chebyshev's inequality. Indeed, by Proposition (i) the variance of $\overline{Q}_n$ and $\overline{W}_n$ are $var\ \overline{Q}_n = o(1)\log_{\omega}^2 n$ and $var\ \overline{W}_n = o(1)\log^2 n^{1/\theta}$, respectively. Then, by Chebyshev's inequality

$$Pr\left\{ \left| \frac{\overline{W}_n}{E\overline{W}_n} - 1 \right| > \varepsilon \right\} \le \frac{var\ \overline{W}_n}{\varepsilon^2 (E\overline{W}_n)^2} = o(1) \tag{2.10}$$

The same is true for $\overline{Q}_n$. Hence (2.10) immediately implies (2.9), and in the next part of this section we only concentrate on proving Proposition (i). The plan for doing that is the following: in the next subsection we show how to establish an upper bound on the maximum of the waiting time , and then we prove a tight lower bound. To simplify our discussion, we shall deal mainly with the waiting time $\overline{W}_n$ knowing that the maximum queue length $\overline{Q}_n$ can be derived in a similar way.

2.2 Upper Bound

The upper bound on the maximum queue length and waiting time derive easily from the following result of Lai and Robbins [LR1, LR2], which is slightly generalized in the next lemma (see also [SZ1, BMS]).

Lemma 1. Let $Y_1, Y_2, \ldots, Y_n$ are identically distributed nonnegative random variables with distribution function $G(y)$. Denote $M_n = \max_{1 \le k \le n} Y_k$.

(i) The following bound holds for the r-th moment EM_n^r of M_n

$$EM_n^r \le a_n + n \int_{a_n}^{\infty} [1 - G(y^{1/r})]dy \tag{2.11}$$

where a_n is the smallest solution of

$$n[1 - G(a_n^{1/r})] = 1 \tag{2.12}$$

(ii) If $Y_1, Y_2, \ldots, Y_n$ are independent random variables satisfying, in addition, the following constraints

$$EY_1^p < \infty \quad \text{for some } p \tag{2.13a}$$

$$G(y) < 1 \quad \text{for} \quad y < \infty \tag{2.13b}$$

$$\lim_{y \to \infty} \frac{1 - G(cy)}{1 - G(y)} = 0 \quad \text{for all} \quad c > 1 \tag{2.13c}$$

then

$$\lim_{n \to \infty} \frac{EM_n^r}{a_n^r} = 1 \tag{2.14}$$

that is, $EM_n^r = a_n^r(1 + o(1))$ for large n. In fact, a_n in (2.14) is merely a solution of (2.12) with $r = 1$, that is,

$$n[1-G(a_n)]=1 \tag{2.12a}$$

Proof. To prove Lemma 1(i), we first note that $EM_n^r = E\max\{Y_1^r, Y_2^r, \dots, Y_n^r\}$ and $Pr\{Y_1^r < y\} = Pr\{Y_1 < y^{1/r}\} = G(y^{1/r})$. Then, as in Lai and Robbins [LR1], one easily shows the following

$$M_n^r \le a + \sum_{k=1}^{n} [Y_k^r - a]^+ \tag{2.15}$$

where a is any number and $x^+ = \max\{0,x\}$. Taking the expectation of (2.15), we obtain

$$EM_n^r \le a + n\int_a^\infty [1 - G(y^{1/r})]dy, \tag{2.16}$$

and finally, minimizing the RHS of (2.16) with respect to a leads to a_n defined in (2.12). Part (ii) of the Lemma 1 is proved in [LR2].

∎

To prove upper bounds on $\overline{W}_n = \max_{1\le k\le n} W_k$ and $\overline{Q}_n = \max_{1\le k\le n} Q_k$, we use Lemma 1(i) and our estimates (2.3) and (2.5) on the tail of the queue length and the waiting time distributions. Let us focus on the waiting time. For stable systems, i.e., $\rho < 1$, a stationary distribution of W_k exists, hence $\{W_k\}_{k=1}^n$ are identically distributed, hence Lemma 1(i) can be applied, and without loosing generality we can assume $r=1$. To compute a_n from (2.12a) and our tail estimate (2.3), we note that there exists $\varepsilon > 0$ tending to zero for $n \to \infty$ such that [BMS]

$$a_n = \frac{\log nc_1(1+\varepsilon)}{\theta} \tag{2.17a}$$

In the case of queue length a_n becomes

$$a_n = \frac{\log nc_2(1+\varepsilon)}{\log \omega^{-1}} \tag{2.17b}$$

Applying the above to (2.16), one immediately proves that the second part (i.e., the integral part) of the RHS of (2.16) is O(1). Therefore,

$$E\overline{W}_n^r \le a_n^r + O(1) \tag{2.18a}$$

$$E\overline{Q}_n^r \le a_n^r + O(1) \tag{2.18b}$$

where a_n is given in (2.17a) for waiting time, and in (2.17b) for the queue length.

2.3 Lower Bound

Here is the strategy to attack our problem. Let L_n denote the number of busy periods completed just prior to the arrival of the n-th customer. We shall prove that $L_n \sim n\alpha$ *almost surely* (a.s.) for some constant α. Let a random customer arriving *during the k-th busy period* sees U_k customers in the queue. Busy periods in GI|G|1 are i.i.d. random variables, hence $\{U_n\}_{k=1}^{L_n}$ is i.i.d. sequence of random variables. Naturally

$$\overline{Q}_n \geq \max_{1 \leq k \leq L_n} U_k \tag{2.19}$$

where '$\geq$' means "stochastically greater" [STO]. Let $M_{L_n} = \max_{1 \leq k \leq L_n} \{U_k\}$. In particular, (2.19) implies that $E\overline{Q}_n^r \geq EM_{L_n}^r$ for any $r > 0$. To compute EM_{L_n} we first prove $L_n \sim n\alpha$ *a.s.* which directly implies that $EM_{L_n} \sim EM_{n\alpha}$. The random variables U_k are i.i.d. and one can choose them in such a way (see below) that they admit the same tail behavior as Q_k (see (2.5)). We shall use Lemma 1(ii) to show that $EM_{n\alpha} \sim \log_\omega(n\alpha)^{-1}$. This, and our upper bound (2.18) derived above, will imply Proposition 1(i) formula (2.8a), and simple manipulation will lead to (2.8b).

The plan just described needs to be accompanied with a proof of some technicalities which follow. We first consider G|G|1 queueing system, and show that

Lemma 2. For large n and $\rho < 1$, $L_n \sim n\alpha$ *almost surely,* where α^{-1} is the average number of customers served in a busy periods.

Proof. The proof follows, with some slight changes, from the proof of Lemma 1 in Heyde [HEY], and it is a simple consequence of the renewal theorem. For the completeness, we provide a sketch of the proof. Let D_n be the number of customer servers in the first n busy periods. Then $D_n = \sum_{i-1}^{n} V_i$ where V_i is the number of customers served in the i-th busy period. Since V_i are i.i.d. [ASM], and letting $\alpha^{-1} = EV_i > 0$, then by strong law of large numbers, $W_n/n \sim \alpha^{-1}$ *a.s.* But, with details found in [GAL, BAN], we note that $D_{L_n}/L_n \sim n/L_n \sim \alpha$ *a.s.*, hence $L_n \sim n\alpha$ *a.s.*

■

The next step is to show that $EM_{L_n} \sim EM_{n\alpha}$ for large n. Indeed without care of details, we can easily estimate

$$Pr\{M_{L_n} > x\} = \sum_{\ell=1}^{\infty} Pr\{U_1 > x \text{ or } U_2 > x \text{ or}, \ldots, \text{ or } U_\ell > x\}\; Pr\{L_n = \ell\}$$

$$= \sum_{\ell=1}^{n\alpha-\varepsilon} P(\cdot) + \sum_{\ell=n\alpha-\varepsilon}^{n\alpha+\varepsilon} P(\cdot) + \sum_{\ell=n\alpha+\varepsilon}^{\infty} P(\cdot)$$

where $P(\cdot)$ is the summand from the first line of the above. Since $L_n \sim n\alpha$ for $n \to \infty$, hence the first and the third sums tend to zero for $n \to \infty$. Then, noting $EM_{L_n} = \sum_{x=0}^{\infty} Pr\{M_{L_n} > x\}$, one proves immediately $EM_{L_n} \sim EM_{n\alpha}$ (for more detailed proof see [GAL, Sec. 6.2]).

To complete the proof, we note that for some K, $M_K = \max_{1 \leq k \leq K} U_k$ can be estimated by Lemma 1(ii). Indeed, U_k are i.i.d. and let U_k be selected in such a way that it represents maximum queue length in the k-th busy period. Then, Iglehart [IGL] has shown that for G|G|1 U_k admits the same tail distribution as Q_n, so one immediatley proves that $\overline{Q}_n \geq (1+o(1)) \cdot \log_\omega n^{-1}$. This and (2.5) imply that conditions (2.13) hold, so applying Lemma 1(ii) one immediately proves that $M_K \sim \log K / \log \omega^{-1}$. Putting everything together, we have shown

$$EM_{L_n} \sim EM_{n\alpha} \sim \frac{\log n\,\alpha}{\log \omega^{-1}} \tag{2.20}$$

For PH|PH|c queue with c servers the proof of the upper bound repeats the same arguments as above but the tail distribution (2.6) is used. For the lower bound we reduce the problem to G|G|1, for which we have just established the asymptotics. Let us define a G|G|1 queueing system that has exactly the same arrival process as our PH|PH|c system under considerations, and exactly the same service distribution except that the server in the G|G|1 system works c times faster. In other words, the service density function in the G|G|1 is $c \cdot f(cx)$ where $f(x)$ is the density function of the service time in the PH|PH|c queue. Naturally, the Laplace-Stieltjes transform of the service times in G|G|1 is $B^*(s/c)$, and hence the solution ω of (2.7b) is the same for the constructed G|G|1 and PH|PH|c queues. But, we know (cf. [[KLE]) that the queue length in the constructed G|G|1 system is stochastically smaller than in the PH|PH|c system, hence using just proved results for single server systems we show that the queue length in PH|PH|c is bounded from the below by $(1+o(1)) \cdot \log_{\omega} n^{-1}$, and this completes the proof of our Proposition.

3. APPLICATIONS

This section discusses in a more detailed fashion one possible application of our main result from Section 2, namely to *hashing with lazy deletion,* which was introduced by Van Wyk and Vitter in [WV], and carefully analyzed by Morrison et al. [MSW], (see also [AHU]).

Here is a sketch of the model description of the hashing with lazy deletion. For more details, the reader should consult [MSW, WV]. We quote from [MSW]: "A sequence of items is given; each item includes a search key, a starting time and an expiration time. The items arrive in the order of their starting times and each item must be kept in a dynamic dictionary (available for searching), until the arrival of an item whose starting time is later than the items expiration time". Two quantities are of interest in such a model, namely, N_t being the number of items that start at or before time t and expire at or after time t, and the actual number of items kept in the dictionary. If H is the number of buckets in the hashing, then by $U_{t,H}$, we denote the actual number of items present at time t. Naturally, $N_t \le U_{t,H}$ for all H.

The model just described can be rephrased in the queueing terminology as follows. We consider GI|G|∞ queueing system. Under FIFO discipline, N_t represents the number of items in GI|G|∞, while $U_{t,H}$ can be interpreted as the number of customers in a system in which only an arriving customer can free the expired (served) customers. In a more descriptive way, the hashing with lazy deletion can be viewed as a queueing system with a gate (a door), which is open *only* by arriving customers. Customers who have completed their services must wait for an arriving customer before they can leave the system.

Our purpose is to analyze $\max_{1 \le t \le n} \{N_t\}$ and $\max_{1 \le t \le n} \{U_{t,H}\}$. To compare our results with those obtained in [MSW], we adopt the same assumptions, that is, we restrict our analysis to M|M|∞ queueing systems with the arrival rate λ and the service rate μ, however our results work for M|G|∞ system too. At the beginning, we assume $H = 1$, since for any H the following holds [MSW]

$$U_{t,H} = \sum_{i=1}^{H} U_{t,1}^{(i)}(\lambda/H) \tag{3.1}$$

where $U_{t,1}^{(i)}(\lambda/H)$ is the number of customers in a M|M|∞ queueing system with $H = 1$ and with Poisson arrival rate λ/H. We return to (3.1) later, and we shall use notation U_t rather than $U_{t,1}$ as long as it does not cause a confusion.

Let $H = 1$ (single bucket) and consider two M|G|∞ systems: one with FIFO discipline and the other describing the hashing system with lazy deletion. Then, for the first system under stationary assumption [KLE]

$$p_j = Pr\{N_t = j\} = \frac{\rho^j}{j!}\, e^{-\rho} \tag{3.2}$$

where $\rho = \lambda/\mu$. In [MSW] it was proved that for M|M|∞ the appropriate distribution for U_t is given by

$$P_j = Pr\{U_t = j\} = p_{j-1} \quad j \geq 1 \tag{3.3}$$

In a forthcoming paper we prove that (3.3) holds also for M|G|∞. In addition, we shall prove that for G|M|∞ (3.3) should be replaced by

$$\rho P_j = j\, p_j\, .$$

In this paper, however, we confine to M|G|∞ queueing systems. Using (3.2) and (3.3) we estimate $\bar{U}_n = \max\limits_{1 \leq t \leq n} U_t$ and $\bar{N}_n = \max\limits_{1 \leq t \leq n} N_t$. The methodology from Section 2 applies without any significant changes, however, one must consider limiting behavior of ω as $c \to \infty$. In particular, we conclude that both $\bar{N}_n$ and $\bar{U}_n$ are asymptotically equal to the root a_n of the equation (2.12a) in Lemma 1.

At first, we derive directly (i.e., without appealing to our Proposition) an estimate on the maximum queue length $\bar{N}_n$ in the M|G|∞ system with FIFO discipline. The complement of the distribution function for N_t can be computed as follows [REN]

$$1 - F(x) = Pr\{N_t > x\} = \sum_{j=x}^{\infty} \frac{\rho^j}{j!}\, e^{-\rho} = \frac{\gamma(x,\rho)}{\Gamma(x)} \tag{3.4}$$

where $\gamma(x,\rho)$ is *the incomplete gamma* function defined as [AS, GAU]

$$\gamma(x,\rho) = \int_0^{\rho} t^{x-1} e^{-t} dt$$

and $\Gamma(x)$ is the gamma function defined as $\gamma(x,\rho)$ [AS]. As in Section 2, we need only the tail of the function $1 - F(x)$. But, [GAU]

$$\frac{\gamma(x,\rho)}{\Gamma(x)} \sim \frac{e^{-\rho}\rho^x}{\Gamma(x+1)} \qquad x \to \infty \tag{3.5}$$

and ρ is bounded. Then by Lemma 1 and the spirit of Section 2 (which is not presented here to avoid repetitiveness), we recognize that $E\bar{N}_n \sim a_n$ where a_n is the smallest solution of the equation (2.12a) of Lemma 1 (ii), that is, the following equation

$$n\, \frac{\gamma(a_n\, , \rho)}{\Gamma(a_n)} = 1 \tag{3.6}$$

We use (3.5) to simplify (3.6) to

$$n\, \frac{\rho^{a_n} e^{-\rho}}{\Gamma(a_n+1)} \simeq 1 \tag{3.7}$$

To solve (3.7) we first apply Stirling's approximation to $\Gamma(a_n+1)$ in order to get

$$n\,e^{-\rho} = \left[\frac{a_n}{\rho\, e}\right]^{a_n} \tag{3.8}$$

Introducing $x = a_n/(\rho e)$ and $N = (n\,e^{-\rho})^{1/\rho e}$, equation (3.8) reduces to

$$x^x = N \tag{3.9}$$

This equation can be easily solved asymptotically, and one proves that the original equation (3.7) possesses the following asymptotic solution

$$a_n = (1+o(1))\cdot\frac{\log n}{\operatorname{loglog} n} \tag{3.10}$$

So we finally we obtain

$$E\overline{N}_n = \frac{\log n}{\operatorname{loglog} n}(1+o(1)) \tag{3.11a}$$

and

$$\lim_{n\to\infty}\frac{\overline{N}_n}{\log n/\operatorname{loglog} n} = 1 \;\; in\ probability \tag{3.11b}$$

Also, in the light of (3.3), one immediately shows that $\overline{U}_n/\overline{N}_n \sim 1$, hence $\overline{U}_n/(\log n/\operatorname{loglog} n) \sim 1$ in probability.

Finally, we investigate the hashing with $H > 1$ buckets. Let $\rho_H = \dfrac{\lambda}{H\mu}$. Then, by (3.1)–(3.3), we note that $U_{t,H}$ is a sum of H independent (truncated) Poisson processes, each with parameter ρ_H. This implies that $U_{t,H}$ is Poisson distributed, too, with parameter $H\rho_H = \rho = \lambda/\mu$ [REN]. Therefore, $Pr\{U_{t,H} > x\} = \gamma(x, H\rho_H)/\Gamma(x) \sim e^{-\rho}\rho^x/\Gamma(x+1)$ for $x \to \infty$, and the same arguments as above lead to

$$E\overline{U}_{n,H} = \frac{\log n}{\operatorname{loglog} n}\cdot(1+o(1)) \tag{3.12a}$$

and

$$\lim_{n\to\infty}\frac{\overline{U}_{n,H}}{\log n/\operatorname{loglog} n} = 1 \;\; in\ probability \tag{3.12b}$$

These results are consistent with the extensive numerical calculations presented in [MSW] for M|M|∞. They show the same rate of growth. However, since we restrict our interest to the leading factor in the asymptotics of $\max\{U_t\}$ and $\max\{N_t\}$, we cannot estimate $E\{\max_{1\le t\le n} U_{t,H} - \max_{1\le t\le n} N_{t,H}\}$. For this we need exact asymptotics up to the second leading term. From our analysis, however, we know that $E\{\max_{1\le t\le n} U_{t,H} - \max_{1\le t\le n} N_{t,H}\} = o(\log n)$. Numerical results reveal that this difference if $O(H)$, and a more careful analysis of the same sort as in our paper, *may* lead to that result. In fact, in a forthcoming paper we shall prove that for any G|G|∞ hashing with lazy deletion and $H > 1$ the following holds

$$E\{\max_{1\le t\le n} U_{t,H}\} - E\{\max_{1\le t\le n} N^*_{t,H}\} = H$$

where $N^*_{t,H}$ represents the length of queues in all buckets found by *arriving customers*.

REFERENCES

[AHU] Aho, A., Hopcroft, J. and Ullman, J., *Data Structures and Algorithms*, Addison-Wesley, Reading, MA (1983).

[AS] Abramowitz, M. and Stegun, I., *Handbook of Mathematical Functions*, Dover, New York (1964).

[ASM] Asmussen, S., *Applied Probability and queues*, John Wiley & Sons, Chichester (1987).

[BMS] Baccelli, F., Makowsky, A. and Shwartz, A., The fork-join queue and related systems with synchronization constraints: Stochastic ordering, approximations and computable bounds, preprint (1987).

[BAN] Barndorff-Nielsen, O., On the limit distribution of the maximum of a random number of independent random variables, *Acta Math. Acad. Sci. Hungrar.*, 15, 399–403 (1964).

[COH] Cohen, J.W., Extreme value distribution for the M|G|1 and the GI|M|1 queueing systems, *Ann. Inst. H. Poincare Sect. B.*, 4, 83–98 (1968).

[FEL] Feller, W., *An Introduction to Probability Theory and its Applications*, Vol. II, John Wiley & Sons, New York (1971).

[GAL] Galambos, J., *The Asymptotic Theory of Extreme Order Statistics*, John Wiley & Sons, New York (1978).

[GAU] Gautschi, W., A computational procedure for incomplete gamma functions, *ACM Trans. on Mathematical Software*, 5, 466–481 (1979).

[HEY] Heyde, C.C., On the growth of the maximum queue length in a stable queue, *Operations Res.*, 44, 423–452 (1971).

[IGL] Iglehart, D., Extreme values in the GI|G|1 queue, *The Ann. Math. Statist.*, 43, 627–635 (1972).

[KLE] Kleinrock, L., *Queueing Systems, Vol. 1*, John Wiley & Sons (1976).

[LR1] Lai, T. and Robbins, M., Maximally dependent random variables, *Proc. Nat. Acad. Sci., USA*, 73, 286–288 (1986).

[LR2] Lai, T. and Robbins, H., A class of dependent random variables and their maxima, *Z. Wahrscheinhich*, 42, 89– 111 (1978).

[MSW] Morrison, J., Shepp, L. and Van Wyk, C., A queueing analysis of hashing with lazy deletion, *SIAM J. Comput.*, 16, 1155–1164 (1987).

[REN] Rényi, A., *Probability Theory*, North-Holland, Amsterdam (1970).

[STO] Stoyan, D., *Comparison Methods for Queues and Other Stochastic Models*, John Wiley & Sons, Chichester (1983).

[SW] Szymanski, T.G. and Van Wyk, C., Space efficient algorithms for VLSI artwork analysis, *Proc. 20th Design Automation Conference*, 734–739 (1983).

[SZ1] Szpankowski, W., Digital data structures and order statistics, in Lecture Notes in Computer Science (eds. F. Dehne, J. Sack, N. Santoro), 382, *Algorithms and Data Structures, R pp. 206-217 (1989).*

[SZ2] Szpankowski, W., (Probably) optimal solution to some problem not only on graphs, Purdue University CSD TR-872, 1989 (submitted to a journal) .

[TAK] Takahashi, Y., Asymptotic exponentiality of the tail of the waiting-time distribution in a PH|PH|c queue, *Adv. Appl. Probab.*, 13, 619–630 (1981).

[WV] Van Wyk, C. and Vitter, J.S., The complexity of hashing with lazy deletion, *Algorithmica*, 1, 17–29 (1986).

On the Construction of Minimum Height Generalized Binary Trees[†]

Venkatraman Viswanathan and Shou-Hsuan Stephen Huang*

Department of Computer Science, University of Houston,
Houston TX 77204-3475

Abstract

Static tree structures such as optimum binary trees, optimum split trees and optimum generalized split trees have been devised to minimize the average access cost of keys stored in these structures. With a view to attain lower average access cost, a generalization of the search function to arbitrary boolean functions was proposed. Though such trees could yield lower average access costs, an efficient algorithm to construct such trees seems difficult to find. In this paper, we prove the NP-hardness of a related problem – building a generalized binary tree of minimum height. This seems to suggest that efficient algorithms to build such optimum trees do not exist.

1. INTRODUCTION

Binary search trees and many of their variants have been studied over the last two decades. The variations include one or more of structural restrictions (AVL trees [1], weight balanced trees [10]), node variations (split trees [11], generalized split trees [6]) and multiway generalizations (B-trees [2], multiway search trees [4]) [7]. The performance of these tree structures have been studied as to their expected uses – dynamic or static. Dynamic structures such as AVL trees, B-trees and their variants provide efficient worst-case performance for deletion, insertion and search operations whereas splay trees [12] provide efficient amortized performances. Static structures include optimum binary trees [8], optimum multiway trees [4], optimum split trees [5] and optimum generalized split trees [6]. These structures are efficient when the keys are accessed over a fixed distribution. Deletion and insertion tend to destroy the optimality of the structure and are expensive. Though it is likely that modifications to these structures could improve this, they have been studied primarily from a static point of view.

An important property which these structures maintain is an ordering of the keys (usually lexicographic). Though this is certainly desirable in certain applications it is not as important in applications such as keyword location. When the keys are weighted, the imposed order places restrictions on the achievable average access cost. As a result generalizations were proposed in [7] of the search function. In the traditional search tree, a search for a key is carried out through a series of lexicographic comparisons with keys stored in the nodes. A generalization of this search function to an arbitrary boolean function was proposed. As a result, such trees tend to be balanced under random insertions if the search function chosen is random. An analysis of the expected total path length of such trees is presented in section 3. One possible application is a hashing index similar to multi-level hashing. The boolean function is essentially a boolean hashing function. Search for a key is carried out through

---

[†] This research was supported in part by Texas Advanced Research Program Grant 1080-ARP

* Part of this work was done while the author held a National Research Council – NASA Research Associateship

a series of comparisons until either the key was found on the search path or a leaf node is reached. The leaves contain pointers to buckets. This sort of a structure is similar to binary tries but the search function is not limited to a specific bit of the key.

Insertion into such a tree can be carried out in a way similar to search trees, but deletion may require a reorganization of the entire tree. These structures are therefore primarily static structures. But devising efficient algorithms for some static applications such as finding trees of minimum weighted path length or even minimum height seems difficult. In this paper we prove the NP-completeness of the decision version of this problem. This gives a strong indication that such efficient algorithms do not exist.

2. GENERALIZED BINARY TREES

Definition 1. *Given a set K of n keys and a boolean function $f : K \times K \to \{0,1\}$, a generalized binary tree $T_f(K)$ on these n keys is a binary tree with the additional property that for all $k \in K$, for all k' in the left subtree of the node with key k, $f(k,k') = 1$ and for all k'' in the right subtree, $f(k,k'') = 0$.*

Given n keys and a boolean function f, it is possible to construct more than one generalized binary tree (GBT). Figure 1 shows two GBTs on seven keys for a given boolean function f. The minimum height generalized binary tree problem is to determine the minimum height of all GBTs $T_f(K)$ on these n keys, for a fixed f. The decision version of this problem (DGBT) is to determine whether it is possible to construct a GBT $T_f(K)$, of height $h \leq H$ for a given H.

Lemma 1. *In a GBT $T_f(K)$, any key $k \in K$ such that $f(k',k) = 0$ for all $k' \neq k, k' \in K$ is in the rightmost path of the GBT.*

Proof. Assume that such a key k is not on the rightmost path. Then there exists a key k' such that k is in the left subtree of the subtree rooted at k'. But no such k' exists. Therefore k is on the rightmost path. []

A well known NP-hard problem is to determine a dominating set which is also an independent set (DIS) of minimum cardinality for a graph G [3]. Given a graph $G = (V,E)$, a *dominating set* of G is a set $V' \subseteq V$ such that for every $u \in V \setminus V'$, there exists a $v \in V'$ such that $(u,v) \in E$. An *independent set* of G is a set $V'' \subseteq V$ such that for all $u,v \in V''$, $(u,v) \notin E$. In the decision version of DIS (DDIS), given a graph $G = (V,E)$ and an integer S, it is required to find a subset V' of V such that V' is a dominating set as well as an independent set and $|V'| \leq S$. We shall show that DGBT is NP-complete by a polynomial-time reduction of DDIS to DGBT. We shall let n denote $|V|$.

Given a graph $G = (V,E)$, construct a graph $G' = (V',E')$ with $V' = V \cup \{v'_i, i = 1 \ldots n, v'_i \notin V\}$ and $E' = E$. Define a boolean function f on $V' \times V'$ such that $f(u,v) = 1$ iff $(u,v) \in E'$. The boolean function is the adjacency relation of G'. Given the set of keys, $K = V'$ and f we can construct a GBT $T_f(K)$ using V' as the set of keys and f as the boolean function. There is a one to one correspondence between a key in $T_f(K)$ and a vertex in G'. For convenience we use these terms interchangeably. The meaning should be clear from the context.

Lemma 2. *For every tree $T_f(K)$ constructed from G' as indicated above, there exists a tree $T'_f(K)$ such that*
(1) $T'_f(K)$ contains the same keys on its rightmost path,
(2) $height(T'_f(K)) \leq height(T_f(K))$,
(3) in $T'_f(K)$, the rightmost path is the longest path.

Proof. Consider the following operation between a parent (corresponding to a vertex of 0 degree in G') and its right son. Move up the left subtree of the son one level. Exchange the parent and its son (Figure 2) . This operation preserves the GBT property. The height of the resulting tree does not increase. From Lemma 1, in T_f, all the vertices of degree 0 are on the rightmost path. Applying the operation exhaustively to $T_f(K)$ gives $T'_f(K)$. In this tree all vertices with non-empty left subtrees have at least n descendants on the rightmost path. Since the left subtree cannot contain more than n vertices, the longest path is the rightmost path. $\square$

Lemma 3. *The set of vertices corresponding to the keys in the rightmost path of $T_f(K)$, V'', forms a dominating set and an independent set for G'.*

Proof. From the construction we see that no two vertices in V'' are adjacent. Therefore V'' is an independent set in G'. Any key not in the rightmost path is in the left subtree of some key in the rightmost path. Therefore, any vertex not in V'' is adjacent to some vertex in V''. Therefore V'' is a dominating set and an independent set for G'. $\square$

Lemma 4. *$V'' \cap V$ is a dominating set and an independent set for G.*

Proof. Only vertices of 0 degree have been added to V to form V' which from Lemma 3 is a DIS for G'. Therefore V'' without these additional vertices is a minimum DIS for G. $\square$

Lemma 5. *Corresponding to every U, a DIS for G, there exists a tree $T_f(K)$ whose rightmost path, amongst all keys corresponding to $v \in V$, contains only those keys corresponding to the vertices in U.*

Proof. A tree $T_f(K)$ can be constructed by forming a chain of all the keys corresponding to vertices in U and the n additional keys in $V' \setminus V$. All other keys k, have some key k' in this chain such that $f(k', k) = 1$. Subtrees formed by these keys can be attached as left subtrees of this chain. $\square$

Given an instance I of DDIS, (G, S), construct an instance I' of DGBT, (K, f, H) as follows. Let $K = V'$, f be the adjacency relation of G' and $H = n + S$.

Theorem 1. *The instance I of DDIS has a solution iff the instance I' of DGBT has a solution.*

Proof. If I has a solution, then G has a DIS U of size less than or equal to S. From Lemma 5, there exists a tree $T_f(K)$ which apart from keys corresponding to U has only keys in $V' \setminus V$ in its rightmost path. From Lemma 2 we know that there is a corresponding tree $T'_f(K)$ having the same vertices on the rightmost path. Moreover the rightmost path is the longest path in $T'_f(K)$. $T'_f(K)$ has a height $\leq S+n$ and is therefore a solution to I. If I' has a solution then there is a tree of height $\leq n+S$. From Lemma 2 there exists such a tree $T_f(K)$ whose rightmost path is its longest path. Consider the vertices on the rightmost path which are also in V. There are not more than S of these. From Lemma 4, these form a DIS for G which is a solution to I. ▯

Theorem 2. *DGBT is NP-complete.*

Proof. Membership in NP is easily established by an algorithm which nondeterministically guesses a GBT and succeeds iff the height condition is satisfied. For NP-hardness, the instance I' of DGBT can be constructed in polynomial time from the instance I of DDIS. From Theorem 1 we have DDIS reduces to DGBT. Since DDIS is NP-hard [3], so is DGBT. ▯

3. EXPECTED PATH LENGTH OF A RANDOM GBT

If the boolean function f is random, i.e., $f(k_i, k_j)$ is 0 or 1 with equal probability for all i and j, then after n insertions, it is highly likely that the resulting tree will be "balanced". In this section we derive an upper bound on $E_n(P)$, the expected value of the total path length P after n insertions.

Let $a_k^{(n)}$ $k \geq 0, n \geq 1$ denote the expected number of nodes at level k after n insertions. The root is considered to be at level 0. Obviously, $a_k^{(n)} = 0$ for $k \geq n$. Then the expected total path length is given by

$$E_n(P) = 1 + \sum_{k=1}^{n-1} (k+1)a_k^{(n)} \tag{1}$$

$a_0^{(n+1)}$ is obviously equal to 1. Consider $a_k^{(n+1)}$ $k \geq 1$. After n insertions, the expected number of nodes at level $k-1$ is $a_{k-1}^{(n)}$ and at level k is $a_k^{(n)}$. Therefore the expected number of empty sons of the nodes at level $k-1$ is $2a_{k-1}^{(n)} - a_k^{(n)}$. The probability that the $(n+1)^{\text{st}}$ key is inserted at any of these $2a_{k-1}^{(n)} - a_k^{(n)}$ positions is $\left(\frac{1}{2}\right)^k$. Therefore $a_k^{(n+1)}$ is given by

$$a_k^{(n+1)} = a_k^{(n)} + \frac{\left(2a_{k-1}^{(n)} - a_k^{(n)}\right)}{2^k} \tag{2}$$

Consider $E_{n+1}(P)$,

$$\begin{aligned}
E_{n+1}(P) &= 1 + \sum_{k=1}^{n} (k+1)a_k^{(n+1)} \\
&= 1 + \sum_{k=1}^{n} (k+1)\left[a_k^{(n)} + \frac{\left(2a_{k-1}^{(n)} - a_k^{(n)}\right)}{2^k}\right] \\
&= 1 + \sum_{k=1}^{n} (k+1)a_k^{(n)} + \sum_{k=1}^{n} \frac{(k+1)a_{k-1}^{(n)}}{2^{k-1}} - \sum_{k=1}^{n} \frac{(k+1)a_k^{(n)}}{2^k} \\
&= E_n(P) + a_0^{(n)} - \frac{(n+1)a_n^{(n)}}{2^n} + \sum_{k=0}^{n-1} \frac{a_k^{(n)}}{2^k}.
\end{aligned} \tag{3}$$

Now, $a_0^{(n)} + \ldots + a_{n-1}^{(n)} = n$ and $a_i^{(n)} \leq 2^i$. Therefore the summation in equation (3) above is bounded above by the value of the summation when

$$a_0^{(n)} = 2^0, \quad a_1^{(n)} = 2^1, \quad \ldots, \quad a_{\lceil \log n \rceil - 1}^{(n)} = 2^{\lceil \log n \rceil - 1}$$

and

$$a_{\lceil \log n \rceil}^{(n)} = n - \left[a_0^{(n)} + \ldots + a_{\lceil \log n \rceil - 1}^{(n)}\right].$$

Therefore

$$\sum_{k=0}^{n-1} \frac{a_k^{(n)}}{2^k} \leq \log(n+1) \tag{4}$$

which yields the following inequality for $E_n(P)$.

$$\begin{aligned}
E_{n+1}(P) &\leq E_n(P) + 1 + \log(n+1) \\
&= E_1(P) + n + \sum_{i=2}^{n+1} \log i \\
&= \log{(n+1)!} + n + 1
\end{aligned}$$

which means

$$E_n(P) \leq \log n! + n \tag{5}$$

Using Stirling's approximation, $E_n(P) \leq n \log n - n(\log e - 1) + \frac{1}{2}\log n + O(1)$.

The same analysis can be used for random digital search trees. The analysis given above is a simpler proof of the asymptotic behaviour than the one in [9].

Using the values for digital search trees [9], the expected path length of a key in a random GBT after n insertions is $\log n - 1.71665$. For binary search trees, assuming all possible permutations of keys are equally likely, the expected path length is $1.386 \log n$. For binary tries, the expected path length is $\log n + 1.33275$ whereas for Patricia trees it is $\log n + 0.33275$.

4. CONCLUSIONS

We have shown the NP-completeness of DGBT. This strongly suggests that other problems such as finding a GBT with minimum weighted path length and finding optimum generalized binary split trees [7] are also difficult to solve. One of the drawbacks of a GBT is the high cost of deletion. This renders the data structure as primarily static. Including a split value in nodes apart from the keys makes this problem easier to solve. In such structures, deleting a key does not involve a re-organization of the entire tree.

A related problem is given a set of keys K and a boolean function f, to find the height of a GBT $T_f(K)$ after a worst case sequence of insertions. This is also NP-hard. A polynomial time reduction from the maximum independent set problem can be constructed. Recall that, in the reduction of DDIS to DGBT we constructed a graph G' from G by adding n new vertices of degree 0. Instead of this, we construct a directed graph G'' by (i) replacing every undirected edge (v_i, v_j) in G by two directed edges (v_i, v_j) and (v_j, v_i) and (ii) adding $n+1$ new vertices of in-degree 0 each of which has n outgoing edges leading to the original n vertices. It can be shown that interpreting the vertex set of G'' as the set of keys K and its adjacency relation as the boolean function f, a tree $T_f(K)$ of maximum height has a height of $n + H + 1$ iff G has a maximum independent set of size H. This proves the NP-hardness of the problem.

As a result of the proof presented, it is obvious that restricting the boolean function to symmetricity alone does not help. When the boolean function defines an ordering of the keys we have binary search trees, for which an optimum structure can be built in $O(n^2)$ time [8]. An obvious question to ask is as to what sort of a boolean function makes the problem tractable.

REFERENCES

[1] Adelson-Velskii, G.M. and Landis, E.M., "An algorithm for the organization of information", *Soviet Math. Dokl. 3,* (1962), 1259–1262.

[2] Bayer, R. and McCreight, C., "Organization and maintenance of large ordered indexes", *Acta Informatica 1,* (1972), 173–189.

[3] Garey, M.R. and Johnson, D.S., *Computers and Intractability: A guide to the theory of NP-completeness,* (1979), W.H. Freeman, San Francisco.

[4] Gotlieb, L., "Optimal multiway search trees", *SIAM Journal of Computing 10,* 3 (1981), 17–24.

[5] Hester, J.H., Hirschberg, D.S., Huang, S.-H.S. and Wong, C.K., "Faster construction of optimal binary split trees", *Journal of Algorithms 7,* (1986), 412–424.

[6] Huang, S.-H.S. and Wong, C.K., "Generalized binary split trees", *Acta Informatica 21,* (1984), 69–79.

[7] Huang, S.-H.S., "On the definition and generalization of binary search trees", Technical report #UH-CS-87-10, Dept. Comp. Sc., University of Houston, (1987).

[8] Knuth, D.E., "Optimum binary search trees", *Acta Informatica 1,* (1971), 14–25.

[9] Knuth, D.E., *The art of computer programming, Vol. 3,* (1973), Addison–Wesley, Reading, Mass.

[10] Nievergelt, J. and Reingold, E.M., "Binary search trees of bounded balance", *SIAM Journal of Computing 2,* (1972), 33–43.

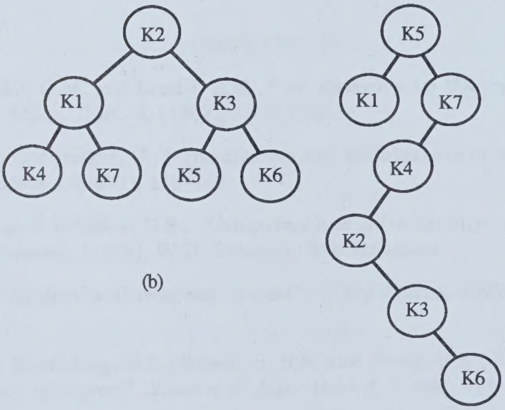
[11] Sheil, B.A., "Median split trees: a fast lookup technique for frequently occurring keys", *Comm. ACM 21,* (1978), 947–958.

[12] Sleator, D.D. and Tarjan, R.E., "Self-adjusting binary search trees", *Journ. ACM 22,* 2 (1985), 652–686.

The boolean function f

	K1	K2	K3	K4	K5	K6	K7
K1	0	0	0	1	0	0	0
K2	1	0	0	1	0	0	1
K3	0	0	0	1	1	0	1
K4	1	1	1	0	1	1	1
K5	1	0	0	0	0	0	0
K6	1	1	1	1	1	0	1
K7	1	1	1	1	1	1	0

(a)


K2
K1
K3
K4
K7
K5
K6
K5
K1
K7
K4
K2
K3
K6

(b)

(c)

Figure 1. Two GBTs with seven keys for the boolean function f

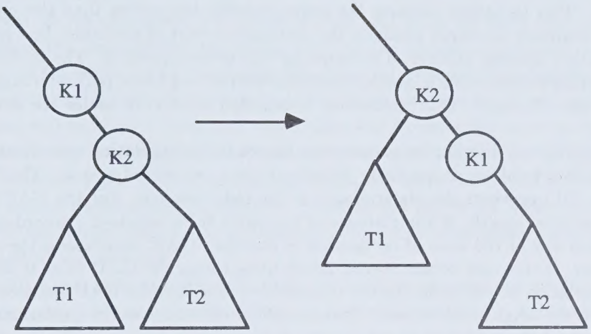

K1
K2
T1
T2
K2
K1
T1
T2

Figure 2. Shifting down a vertex of degree 0.

INTERPOLATION SEARCH FOR ALPHABETIC TABLES

YEHOSHUA PERL
LOIZOS GABRIEL
Institute of Integrated Systems, C.I.S. Department
Newark, New Jersey 07102

One of the fastest search techniques for uniformly distributed sorted numerical tables is interpolation search. This technique accesses the most probable key rather than the middle key as in binary search and continues to search similarly the appropriate part of the table. In a previous work [PIA] we proved a lg lg n average number of accesses for interpolation search. The inefficiency of interpolation search for an alphabetic table is demonstrated by Burton and Lewis [BL] who suggest a robust variation to improve the efficiency. This inefficiency is expected since such tables are usually far from uniform distribution.

Actually the uniform distribution assumption causes the interpolation search, when applied to nonuniformly distributed tables, to partially deteriorate to a sequential search. The GAP modification introduced in [BL] prevents this deterioraion in the following way. Let the GAP be the square root of the search interval length. If the distance of the entry to be accessed , according to the interpolation formula, from any of the ends of he interval is less than GAP, then access the entry of disance GAP from this end. If this case occurs several consecutive times, the GAP value is doubled each time until a real interpolation access occurs or the required key is within the GAP distance and the GAP is reset according to the original definition. This modification improves the performance.However it is still higher than the lg lg n performance for uniform distribution.

The important question is : Can we modify interpolation search to work for nonuniform distribution with a similar performance to the one it demonstrates for uniform distribution ? We can use the property from Probability Theory that applying the cumulative distribution function F to the keys of the table according to this distribution transfers the table into a uniformly distributed table.

To help us in this task we look now into Arithmetic Coding(eg. [WNC]). This is a technique for data compression which maps a string of characters to an interval which is contained in the [0 , 1) interval. This maping is based on the probabilities of the different characters in the coded text either known in advance or accumulated during the encoding.

We can use the same idea of maping a string into a partial interval of the [0, 1) interval according to the probability of the characters to transform the non-uniform distribution of the strings into a uniform distribuion.For each string we apply the arithmetic coding to obtain an interval and the middle point of this inteval is taken as the cumulaive distribution function F for this string. Thus we have a modification of interpolation search for alphabetic tables which should have an average number of lg lg n accesses. We name this variation the Arithmetic Interpolation Search.

We need to define some probaility terms before presenting the theorem. Let () denote the i-th character in the string and let $(,j)$ denote the substring of containing the characters in the positions $, + 1, ..., j$ of . Thus $(1, j)$ denotes the prefix of j characters of .

Let $P1(a)$ denote the probability of the first character in a string to be a. Let $Q1(a)$ denote the probability of the first character in a string to proceed a in the alphabetic order. Let P(a,i,t) denote the conditional probability of a character a in a position of a string which has a prefix t. Let Q(a,i,t) be the conditional probability of a character preceding a (in the alphabetic order) in a position of a string which has prefix t. Let $R(t)$ denote the probability of a string in the file to have a prefix t of j characters Then

$$R(t) = P1(t(1)) \times \prod_{i=2}^{j} P(t(), , t(1, -1))$$

(Note that $R(t)$ is the length of the interval representing the prefix t in the arithmetic coding process). Now given the appropriate probabilities the computation of the arithmetic code of a string of k characters is

$$A() = Q1((1)) + \sum_{i=2}^{k} R((1, -1)) \times Q((), , (1, -1))$$

Theorem: Given the appropriate probabilities for a given file of strings, the arithmetic coding A(s) of a string as defined above is the cumulative distribution function $F()$ for the given file.

To verify the theoretical results we used for our experiments a file containing the alumni list of N.J.I.T. This list contains about 26,000 names, sorted alphabetically. To measure the performance we searched every 50^{th} name giving a total of 512 names of the file.

We calculate separately the probabilities of the first character in each string. For the second position in the string we calculate the probability of the character dependent on the first character in the string. Thus for the probabilities for the second position we need a table of k^2 entries where k is the number of characters in our alphabet. Similarly for the th position we shall need a table of k^i entries.This is beyond the available space resources.

Thus we decided to treat all the characters from position three and on in the same way as follows.For each character we collect the probabilities depending only on the previous character in the string, rather than depending on the whole prefix preceeding the character, without regarding the position of the character. This way the extra table of dependent probabilities will require only k^2 entries. The R conditional probabilities will be also be computed in a similar way by using the same probabilities.

This way we only approximate the conditional probabilities which appear in the formula for arithmetic coding. Nevertheless we believe this is a reasonable approximation since the distribution of the characters in the first and second position are different from those for the rest of the positions and are also the most critical part in the computation of the arithmetic coding. The average and maximum number of accesses are 6.865 and 87, respectively.

Therefore we should realize that we used arithmetic coding as much as our resources enable us. Now we should look for other techniques to avoid the deterioration into a sequential search due to the clustering. As a matter of fact the GAP method described earlier was designed in [BL] for this purpose. Thus we apply the GAP modification with the arithmetic coding to the file. The average and maximum number of accesses, are 6.420 and 14 respectively, a small improvement in the average and a drastic improvement in the maximum number. However looking at specific searches we realize that while the GAP modification shorten considerably the long searches it also lengthen many short searches.

Thus if we want to apply the GAP method we need to do it more carefully so we do not lengthen the short searches where the interpolation search is very efficient.

To achieve this we tried experimentally different formulas.The best result was obtained for the square root of the search interval divided by lg n. We call this variation the SMALLER GAP variation.The average and the maximum number are 5.381 and 18. We see a significant improvement in the average. Looking still for an extra improvement we realize that the very short searches still got longer using the SMALLER GAP method.This is when the access close to the end of the interval is the right one or almost the right one and the modification of the SMALLER GAP variation is lenghtening the searches unnecesarily.The problem is how to distinguish between the cases that the access is near the end of the interval because the arithmetic coding is not exact and the cases where the access is very close to the searched index while both the arithmetic coding and interpolation search are exact.

We found the following variation which distinguishes between these two cases successfully, most of the time. The idea is that if the difference between the calculated index and the end of the interval is less than EPSILON for a very small EPSILON then we should access the key at the end of the interval rather than apply the SMALLER GAP method. We call this method the EPSILON SMALLER GAP variation. For example for EPSILON $= 0.1 * 10^{-8}$ the average and maximum number of accesses are 4.916 and 19, respectively.

This last version is close to the lg lg n performance. Thus achieving our goal of providing a lg lg n method of interpolation search for alphabetic files.

REFERENCES:

[BL] Burton F.W. and Lewis G.N., *A robust variation of interpolation search*, Information processing Letters 10, 1980, pp. 198-201.

[PIA] Perl Y., Itai A.,and Avni H., *Interpolation Search -A Lg Lg N Search*, CACM, 21(1978), 550-553.

[WNC] Witten I.H.,Neal R.M. and Cleary J.G., *Arithmetic coding for data compression*, CACM, 30(1987) 520-540.

AN EFFICIENT ALGORITHM FOR A SPECIAL CASE OF THE SET PARTITION PROBLEM

ALAN JACKSON AND PRAKASH RAMANAN *
Department of Computer Science
University of California
Santa Barbara, CA 93106

ABSTRACT

In this paper, we study the Two Function Set Partition Problem, which is defined as follows: Given a set S of n elements, functions f_1 and f_2 from S to S, and an initial partition $B = (B_1, B_2, ..., B_s)$ of S, find the coarsest refinement $E = (E_1, E_2, ..., E_t)$ of B such that for each i, $i = 1, 2$, and each j, $1 \le j \le t$, $f_i(E_j) \subseteq E_k$ for some k. For the special case when f_1 consists of a single cycle, we present an $O(n\beta(n))$ algorithm, where $\beta(n)$ is the number of distinct prime factors of n. $\beta(n)$ is $\ln\ln n + o(\ln\ln n)$ for almost all n, and is $\Theta(\log n/\log\log n)$ in the worst case.[1] This algorithm represents an improvement over the previously known $O(n\log n)$ algorithm.

1. INTRODUCTION

In this paper, we study the m-Function Set Partition Problem (m-FSP) which is defined as follows (see [1, pp. 157-162]). Let S be a set of n elements, and let f_i, $1 \le i \le m$, be functions from S to S. Given an initial partition $B = (B_1, B_2, ..., B_s)$ of S, find the partition $E = (E_1, E_2, ..., E_t)$ of S with the smallest number of parts, such that

(i) E is a refinement of B; *i.e.*, each E_j is a subset of B_k, for some k, and
(ii) For each i, $1 \le i \le m$, and each j, $1 \le j \le t$, $f_i(E_j) \subseteq E_k$ for some k.

By the Myhill-Nerode theorem [7, 8] (also see [3, pp. 65-67]), E is uniquely determined by S, B and f_i, $1 \le i \le m$. We say that E is the *coarsest refinement* of B that is *consistent* with the f_is. The Single Function Set Partition Problem (SFSP), and the Two Function Set Partition Problem (TFSP) correspond to the cases $m = 1, 2$ respectively.

An important application of this problem arises in minimizing the number of states of a Deterministic Finite Automaton (DFA) (see [1, pp. 143-144] and [3, pp. 16-18]). Hopcroft [2] presented an $O(mn\log n)$ algorithm for m-FSP. For SFSP, Paige, Tarjan and Bonic [9] presented a linear-time algorithm by exploiting several useful properties of the transition graph representing f_1. Unfortunately, these properties do not lead to an $O(mn)$ algorithm for m-FSP. On the other hand, there exists an $O(mn)$ height decision tree that solves m-FSP. Each node of the decision tree contains a query that is of one of two types: (i) What is $f_i(a)$? (ii) Which part B_j contains a?

Note that the degree of a node containing a query of the first (second) type is n (s). Any path from the root of the decision tree to a leaf contains mn queries of type (i) and n queries of type (ii). The solution (*i.e.*, the partition E) for the corresponding instance is stored in the leaf. Unfortunately, this decision tree does not lead to an $O(mn)$ algorithm.

---

* This author's research was supported in part by the National Science Foundation under grant CCR-88-03241

[1] Throughout this paper, ln denotes the natural logarithm, and log denotes logarithm to the base 2.

Let SCSP (for Single Cycle TFSP) denote the special case of TFSP, where the graph of f_1 consists of a single cycle containing all the n elements of S, and f_2 is arbitrary. In Section 2, we briefly describe Paige, Tarjan and Bonic's algorithm for SFSP. In Section 3, we present an $O(n\beta(n))$ algorithm for SCSP, where $\beta(n)$ is the number of distinct prime factors of n. $\beta(n)$ is $\ln\ln n + o(\ln\ln n)$ for almost all n, and is $\Theta(\log n/\log\log n)$ in the worst case.

2. PAIGE, TARJAN AND BONIC'S ALGORITHM

Hopcroft [2] presented an $O(mn\log n)$ algorithm for m–FSP, and Paige, Tarjan and Bonic [9] presented a linear-time algorithm for SFSP. Since our algorithm uses some of the concepts in [9], we briefly describe their algorithm.

Let f be the single function from S to S. Hopcroft presented an $O(n\log n)$ algorithm for SFSP. It can be easily generalized to solve m–FSP in $O(mn\log n)$ time. At any instant, the algorithm maintains a *current partition* $P = (P_1, P_2, \ldots, P_u)$, which is a refinement of B. Partition P is initialized to B, and when the algorithm terminates, it gives the final partition E. The algorithm repeatedly refines the partition until the resulting partition is consistent with f.

The linear-time algorithm of Paige, Tarjan and Bonic for SFSP exploits some nice properties of the graph of f, and cannot be generalized to solve m–FSP efficiently. For any partition $P = (P_1, P_2, \ldots, P_u)$ of S, let the *P–label* of $x \in S$, denoted by $L_P(x)$, be the index of the part in P that contains x. We say that x, $y \in S$ are *equivalent under P*, denoted by $x \equiv_P y$, if x and y are in the same part in P, *i.e.*, $L_P(x) = L_P(y)$. Unlike Hopcroft's algorithm, the algorithm of Paige et al. finds the E-label of the elements, given their B-label and the function f. The basic result they use is the following.

Lemma 2.1. Let $f^0(x) = x$, and $f^i(x) = f(f^{i-1}(x))$, $i > 0$. Then
(i) for all $x, y \in S$, $x \equiv_E y$ iff $x \equiv_B y$ and $f(x) \equiv_E f(y)$
(ii) for all $x, y \in S$, $x \equiv_E y$ iff for all $i \geq 0$, $f^i(x) \equiv_B f^i(y)$ ■

The function $f\colon S \rightarrow S$ represents a directed graph on S, where the out-degree of each element is one. Paige et al. proved the following properties of the graph.

Lemma 2.2.
(i) f must have at least one cycle.
(ii) Each path ends in a cycle.
(iii) No edge can leave any cycle.
(iv) All paths outside of cycles form trees whose roots are in the cycles. ■

By Lemma 2.1, $L_E(x)$ depends only on the B-labels of the elements reachable from x along paths in f. By Lemma 2.2, there are three cases in which x, $y \in S$ can be given the same E-label:
(1) x and y are in the same cycle,
(2) x and y are in different cycles, and
(3) at least one of x and y is in a tree structure

We need the following definitions and notations. Let w be a string of labels. We let $|w|$ denote the length of w. w^i denotes the string consisting of i repetitions of w. For example, $(abc)^3 = abcabcabc$. The *smallest repeating prefix* of a string w, denoted by $srp(w)$, is the smallest prefix v of w such that $v^j = w$ for some integer j.

Let us first consider Case 1. Let C be a cycle with k elements. For $x \in C$, let $LabSeq_x(C)$ be the sequence of labels $(L_B(x), L_B(f(x)), L_B(f^2(x)), \ldots, L_B(f^{k-1}(x)))$. For a given cycle C, let $x' \in C$ be such that $LabSeq_{x'}(C)$ is lexicographically the smallest. Then we let $LabSeq(C) = LabSeq_{x'}(C)$, and let $srp(C) = srp(LabSeq(C))$. If $y = f^i(x')$, for some i, $0 \leq i < k$, then $offset(y)$ is defined to be $i \bmod |srp(C)|$. The equivalences among elements in C are completely determined by $srp(C)$, as stated in the following lemma.

Lemma 2.3. [9] Let C be a cycle with k elements. Then the elements of C can be partitioned into $|srp(C)|$ parts, such that $x, y \in C$ are in the same part if and only if $x \equiv_E y$. For $0 \leq j < |srp(C)|$, the jth part consists of all the elements in C whose offset is j.■

For Case 2, they have the following result.

Lemma 2.4. [9] Let x and y be in cycles C_1 and C_2 respectively. Then $x \equiv_E y$ if and only if $srp(C_1) = srp(C_2)$, and $offset(x) = offset(y)$. ■

For Case 3, Paige et al. note that if $x \equiv_E y$, then the paths from x and y in f must lead to cycles with the same *srp*. Note that each connected component of f contains exactly one cycle, and each node in the cycle is the root of some tree. So, we can first partition S into maximal groups of connected components of f such that all the cycles in any one group have the same *srp*. Any two elements in different groups must have different E-labels. E-labels for the tree vertices can be determined during a root to leaf traversal in accordance with the following rule (see Lemma 2.1): $x \equiv_E y$ iff $x \equiv_B y$ and $f(x) \equiv_E f(y)$.

From now onwards, the algorithm of Paige et al. will be referred to as the *Single Function Refinement* (SFR) algorithm. Given a set S, a function f: $S \rightarrow S$, and an initial partition B of S, the call *SFR*(S, f, B) returns the coarsest refinement of B consistent with f, in linear time.

3. AN EFFICIENT ALGORITHM FOR SCSP

In this section, we present an $O(n\beta(n))$ algorithm for SCSP, where $\beta(n)$ is the number of distinct prime factors of n. $\beta(n) = \ln\ln n + o(\ln\ln n)$ for almost all n, and in the worst case $\beta(n) = \Theta(\log n/\log\log n)$.

First, let us consider the simple *Iterative Refinement* algorithm for TFSP. The algorithm first refines B with respect to f_1, obtaining the partition B'. At any instant, it maintains a current partition P, which is initialized to B'. P is repeatedly refined with respect to f_2 and f_1, until no further refinement is possible. Each refinement with respect to f_1 or f_2 is performed using Paige et al.'s *SFR* algorithm for SFSP. Clearly, the *Iterative Refinement* algorithm produces the correct partition E. Each pass through the *while* loop requires $O(n)$ time. For the TFSP problem, there can be up to $\Theta(n)$ iterations required (see [4]). Hence the algorithm runs in $O(n^2)$ time.

From now onwards, we consider the SCSP problem, where the graph of f_1 consists only of a single cycle C. We let $S = \{1, 2, ..., n\}$, and $f_1(i) = i \bmod n + 1$, $1 \leq i \leq n$. Let $B' = (B'_1, B'_2, ..., B'_l)$ be the coarsest refinement of B that is consistent with f_1. Then $l = |srp(C)|$, and each part in B' has n/l elements.

A sequence $(a + 1, a + 2, ..., a + u)$ of consecutive elements in C is called a ***block***. In this block, for $1 \leq i \leq u$, the ***position*** of $a + i$, denoted by $pos(a + i)$, is i. A block with u elements is also called a *u-block*. If $n \bmod u = 0$, then $\{(au + 1, au + 2, ..., au + u) \mid 0 \leq a < n/u\}$ is called the *u-division of C*. It is easy to prove the following.

Lemma 3.1. If $P = (P_1, P_2, ..., P_u)$ is any refinement of B' that is consistent with f_1,

```
Algorithm Iterative Refinement $(S, f_1, f_2, B)$
begin
        $B' = SFR(S, f_1, B)$
        $P = B'$
        $E = (S)$
        while $E \neq P$ do
        begin
                $E = P$
                $P = SFR(S, f_2, P)$
                $P = SFR(S, f_1, P)$
        end
end
```

then $u \bmod l = 0$ and $n \bmod u = 0$. Also, for each u such that $u \bmod l = 0$ and $n \bmod u = 0$, there is a unique refinement $P = (P_1, P_2, ..., P_u)$ of B' that is consistent with f_1. In this partition P, each part P_i, $1 \leq i \leq u$, must consist of one element from each block in the u-division, all with the same position; without loss of generality, we can let P_i consist of elements with position i. ■

Corollary 3.1.1. If P is any refinement of B' that is consistent with f_1, then all the parts in P must have the same number of elements. In particular, all the parts in E must have the same number of elements. ■

By Lemma 3.1, any refinement $P = (P_1, P_2, ..., P_u)$ of B' that is consistent with f_1 is completely determined by u. We say that the u-division of C is *consistent with* f_2 (or u is *consistent with* f_2), if $f_2(x)$ and $f_2(y)$ have the same position whenever x and y have the same position. It is easy to prove the following.

Lemma 3.2. For u such that $u \bmod l = 0$ and $n \bmod u = 0$, let $P = (P_1, P_2, ..., P_u)$ be the refinement of B' consistent with f_1. Then P is consistent with f_2 iff u is consistent with f_2. ■

Corollary 3.2.1. The number t of parts in E is the smallest integer such that $t \bmod l = 0$, $n \bmod t = 0$, and t is consistent with f_2. ■

For any integer v, let $\alpha(v)$ denote the number of prime factors of v, counting multiplicity. Let $k = n/l$, and $k = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_j^{\alpha_j}$ be the prime factorization of k, where $2 \leq p_1 < p_2 < \cdots < p_j$, and $\alpha_i > 0$, $1 \leq i \leq j$; then $\alpha(k) = \sum_{i=1}^{j} \alpha_i$. We have the following bound on the time complexity of the *Iterative Refinement* algorithm.

Theorem 3.3. The *Iterative Refinement* algorithm solves SCSP in $O(n \log\log n)$ time almost always. In the worst case, it runs in $O(n \log n)$ time.

Proof. Let $P = (P_1, P_2, ..., P_u)$ be the refinement of B', at the end of some pass through the *while* loop. By Lemma 3.1, $u \bmod l = 0$ and $n \bmod u = 0$. So, for each pass through the loop (except the last one), u increases at least by some factor p_i, $1 \leq i \leq j$. So, the number of passes through the *while* loop is at most $\alpha(k) = \sum_{i=1}^{j} \alpha_i$. Since $\alpha(n) \geq \alpha(k)$, the algorithm solves SCSP in $O(n\alpha(n))$ time. $\alpha(n) = \ln\ln n + o(\ln\ln n)$ for almost all n (see [5], [6, pp. 71-79]), and clearly $\alpha(n) = \log n$ in the worst case. ■

For each (n, k, l), there exist instances of SCSP that require $\alpha(k)$ iterations (see [4]). So, the bound of Theorem 3.3 is essentially tight.

Now we shall present a more efficient algorithm for SCSP. The main part of our algorithm is the function *Numparts*. We can obtain t as $t = Numparts(n)$. $Numparts(n)$ considers each prime factor p_i of k one at a time, and makes the call $Consistent(u, n)$, with $u = n/p_i$. u is a *var* parameter (as in Pascal) of the function *Consistent*. $Consistent(u, n)$ checks if u is consistent with f_2. In performing this check, three possible scenarios can arise:

1) u is consistent with f_2. In this case, the value of u is not changed, and the function is assigned the value *true*.

2) u is not consistent with f_2, but some proper divisor of u may be consistent with f_2. In this case, *Consistent* calls itself recursively, and finds the *largest* divisor u' of u that is consistent with f_2. The *var* parameter u is assigned the value of u', and the function is assigned the value *true*.

3) Neither u nor any of its divisors is consistent with f_2. In this case, u is assigned the value n, and the function is assigned the value *false*.

If any of the calls $Consistent(u, n)$ made by $Numparts(n)$ returns the value *true*, then t is the *smallest* divisor of u such that $t \bmod l = 0$ and t is consistent with f_2. t can be found by the recursive call $Numparts(u)$, which only needs to look at the elements $\{1, 2, ..., u\}$.

We let $gcd(v, w)$ denote the greatest common divisor of integers v and w. Also, when the *exit* statement is encountered during execution of a function, the execution of that function is terminated.

To speed up the *gcd* computation in *Consistent*, we initially set up the global array *CommonFacts* of pointers to linked lists, the array indexed from 1 to k. $CommonFacts[v]$ points to $gcd(v, k)$, stored in the form of a linked list. Each record in the linked list consists of two fields: a *prime* and its largest *power* that fits in the *gcd*. The factors are stored in decreasing order of *primes* (*e.g.*, for $k = 48$,

```
function $Numparts(n)$: integer
/* $k = n/l,\ k = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_j^{\alpha_j}$ */
begin
    for $i$=1 to $j$ do
        $u = n/p_i$
        if $Consistent(u, n)$
          then
                $Numparts = Numparts(u)$
                exit
    endfor
    $Numparts = n$
end
```

```
function Consistent(var u; n): boolean
begin
    $u' = u$
    for $b = 1$ to $u$ do
        for $a = 1$ to $n/u - 1$ do
            if $|f_2(b) - f_2(b + au)| \bmod l \neq 0$
              then
                  $u = n$
                  Consistent = false
                  exit
              else
                  if $|f_2(b) - f_2(b + au)| \bmod u' \neq 0$
                    then $u' = gcd(u', |f_2(b) - f_2(b + au)|)$
        endfor
    endfor
    if $u' = u$
      then Consistent = true
      else
          Consistent = Consistent(u', u)
          $u = u'$
end
```

$CommonFacts[36] = 3 \rightarrow 2^2$). We have the following results.

Lemma 3.4. The prime factorization of k can be computed in $O(n)$ time.

Lemma 3.5. The array *CommonFacts* can be constructed in $O(n\beta(n))$ time.

Proof. We can construct the array by successively considering each prime power in the prime factorization of $k = {p_1}^{\alpha_1}{p_2}^{\alpha_2} \cdots {p_j}^{\alpha_j}$. Consider the p_i's in increasing order, *i.e.*, i ranging from 1 to j. For each p_i do the following: for all integers r, $1 \leq r \leq k/p_i$, insert a record at the head of the linked list for $CommonFacts[r\,p_i]$ with the fields $prime = p_i$ and $power = 1$. If $\alpha_i > 1$, then for c ranging from 2 to α_i, for all d, $1 \leq d \leq k/{p_i}^c$, increment by one the *power* field of the record for $prime = p_i$ in $CommonFacts[d\,{p_i}^c]$. For each p_i, the time required to set the records with $prime = p_i$ is $k/p_i + k/{p_i}^2 + k/{p_i}^3 + \cdots = O(k)$. The total time for all the primes is $O(k\beta(n))$. ■

Lemma 3.6. $Consistent(u, n)$ runs in time $CT(u, n) = O(n)$.

Proof. The body of the inner *for* loop will be executed at most n times. Except for the *gcd* computation, each statement in the body takes $O(1)$ time. For the *gcd*, we can compute gcd/l by simultaneously traversing the linked lists at $CommonFacts[u'/l]$ and $CommonFacts[|f_2(b) - f_2(b + au)|/l]$, building gcd/l from the prime factors that the two lists have in common. Since both the lists are in decreasing order of primes, this can be performed in one traversal of the lists, in $\beta(n)$ time. After each *gcd* computation, u' decreases at least by some prime factor p_i. Therefore the *gcd* computation will be performed at most $\alpha(n)$ times, for a total time complexity of $O(\alpha(n)\beta(n)) = O(n)$.

Therefore *Consistent*(u, n) runs in time $CT(u, n) \leq cn + CT(u', u)$, where $u \leq n/p_1$, $u' \leq u/p_1$. Hence $CT(u, n) = O(n)$. ■

Lemma 3.7. *Numparts*(n) runs in time $NP(n) = O(n\beta(n))$.

Proof. The body of the *for* loop is executed at most $\beta(n)$ times. *Consistent*(u, n) runs in $O(n)$ time. If the *then* section is encountered, then the function terminates after the recursive call. Hence *Numparts*(n) runs in time $NP(n) \leq \beta(n)O(n) + NP(u)$, where $u \leq n/p_1$. Hence $NP(n) = O(n\beta(n))$. ■

Theorem 3.8. SCSP can be solved in $O(n \log\log n)$ time almost always. In the worst case, it can be solved in $O(n \log n / \log\log n)$ time.

Proof. By Lemma 3.7, SCSP can be solved in $O(n\beta(n))$ time. $\beta(n) = \ln\ln n + o(\ln\ln n)$ for almost all n (see [5], [6, pp. 71-79]), and $\beta(n) = \Theta(\log n / \log\log n)$ in the worst case. ■

REFERENCES

1. A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Mass., 1974.
2. J.E. Hopcroft, An $n \log n$ Algorithm for Minimizing States in a Finite Automaton, in *Theory of Machines and Computation,* Kohavi and Paz, eds., Academic Press, New York, 1971, pp. 189-196.
3. J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation,* Addison-Wesley, Reading, Mass., 1979.
4. A. G. Jackson, An Efficient Algorithm for a Special Case of the Set Partition Problem, MS thesis, Department of Computer Science, University of California, Santa Barbara, Calif., July 1989.
5. Mark Kac, *Probability Theory and Related Topics in the Physical Sciences: A Seminar in Applied Mathematics,* arranged by American Mathematical Society, Boulder, Colo., June 1957.
6. Mark Kac, *Statistical Independence in Probability, Analysis and Number Theory,* The Carus mathematical monographs, vol. **12**, The Mathematical Association of America, 1959.
7. J. Myhill, Finite Automata and the Representation of Events, WADD TR-57-624, Wright Patterson AFB, Ohio, 1957.
8. A. Nerode, Linear Automata Transformations, *Proc. AMS* **9** (1958), pp. 541-544.
9. R. Paige, R.E. Tarjan and R. Bonic, A Linear Time Solution to the Single Function Coarsest Partition Problem, *Theoret. Comp. Sci.* **40** (1985), pp. 67-84.

A LINEAR TIME ALGORITHM FOR THE TWO TRACK ASSIGNMENT PROBLEM

WEN–LIAN HSU AND KUO–HUI TSAI
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 60208; and Institute of Information Science, Academia Sinica, Republic of China.

Abstract. Given two interval tracks T_1, T_2 and a set of intervals, the two track assignment problem is to assign a maximum number of intervals to the tracks such that all intervals assigned to T_i are non–overlapping and contained in T_i for $i = 1, 2$. This problem is closely related to the two independent set problem on circular–arc graphs. Assuming the interval endpoints are presorted, we present $O(n)$ time algorithms for both problems in this paper. This represents a substantial improvement over a proclaimed $O(n^3)$ algorithm.

1. INTRODUCTION

Consider a scheduling problem in which there are n jobs and two machines. Each job has specified start time, finish time and can be assigned to at most one machine (once assigned, must be processed from the start time continuously to the finish time). Each machine can operate on at most one job at a time. Suppose each machine has a certain operational period (interval track). The objective is to determine an assignment of jobs to machines such that the number of completed jobs is maximized.

More formally, the problem can be defined as follows. Given two interval tracks (a_1,b_1) and (a_2,b_2) (representing the operational periods of the two machines) and a family H of intervals, the ***two track assignment*** (TTA) ***problem*** on H is to determine two disjoint subsets R_1 and R_2 in H such that

(a) all intervals of R_i are contained in (a_i,b_i) for $i = 1, 2$.

(b) no two intervals of R_i overlap with each other for $i = 1, 2$.

(c) $|R_1 \cup R_2|$ is maximum (this cardinality is denoted by $\tau(H)$).

This problem has previously been considered by Yeh and LaPaugh [11]. They claimed that it can be solved in $O(n)$ time. However, we believe there is a flaw in their algorithm. The TTA problem is closely related to the maximum bipartite subgraph problem on certain graphs discussed below.

Define a graph G to be an ***interval graph*** if there is an interval family H and a one–to–one mapping of the vertices of G and the arcs in H such that two vertices in G are adjacent if and only if their corresponding intervals in H overlap. G is also referred to as the ***intersection graph*** of H. A graph G is a ***circular–arc graph*** if it is the intersection graph of a family F of circular arcs on a unit circle. Let n be the number of arcs in F. Circular–arc graphs are rich in combinatorial structures. Various characterization and optimization problems on interval graphs has been studied [1,2,3,7,9]. For convenience, we shall consider intervals or arcs in the family rather than vertices in its corresponding graph G. An example of circular–arc family is shown in Figure 1.1.

A set of arcs is ***independent*** if no two of them overlap with each other. The ***maximum independent set (MIS) problem*** on an arc family F is to determine a maximum size (denoted by $\alpha(F)$) independent set in F, and the ***maximum two–independent set (MTIS) problem*** on F is to determine two disjoint independent sets S_1 and S_2 in F such that $|S_1 \cup S_2|$ is maximum (denoted by $\alpha_2(F)$). This problem is a special case of the maximum k–independent set problem on general graphs. The latter is NP–hard in general [6,8] and solvable for chordal graphs [10].

In this paper, we show that the TTA problem on families of intervals can be transformed into the MTIS problem on families of circular arcs in $O(n)$ time, and both problems can be solved in $O(n)$ time given that the endpoints have already been sorted. In contrast, Yeh and LaPaugh [11] argued that the MTIS problem on circular–arc graphs can be transformed into $O(\delta^2)$ number of TTA problems, and thus, can be solved in $O(\delta^2 \cdot n)$ time, where δ is the minimum number of arcs passing through some point of the unit circle (hence, the worst case would be $O(n^3)$).

This paper is arranged as follows. Section 2 reviews the results of the MIS problem on circular–arc graph. Properties of optimal two–independent sets are analyzed in section 3. In section 4, an $O(n)$ algorithm is presented for the MTIS problem on circular–arc graphs. Finally, we illustrate the transformation from the two–track assignment problem to the MTIS problem in section 5.

2. REVIEW OF THE MIS PROBLEM ON CIRCULAR–ARC GRAPHS

There are several $O(n)$ algorithms for the MIS problem on circular–arc graphs. We shall briefly describe the one by Hsu and Tsai [5].

Without loss of generality, assume all arc endpoints are distinct and no arc covers the entire circle. Label the n arcs arbitrarily from 1 through n. Starting with any endpoint, label all the endpoints from 1 to 2n according to their clockwise order. Denote by (p,q) the arc i that begins at endpoint p and ends at endpoint q in the clockwise direction. Define p to be the ***head*** of i (denoted by **h(i)**) and q to be the ***tail*** (denoted by **t(i)**).

The continuous part of the circle that begins with an endpoint c and ends with an endpoint d in the clockwise direction is referred to as ***segment* (c,d)** of the circle.

We use "arc" to refer to a member of F and "segment" to refer to a part of the circle between two endpoints. By this definition, an arc (p,q) of F is also a segment (p,q). A point on the circle is said to be in arc (or segment) (p,q) if it falls within the interior of segment (p,q).

An arc i in F is said to be ***contained in*** another arc j if every point of i is contained in arc j. An arc family F is said to be ***proper*** if no arc in F is contained in another. Given any arc family F, let C(F) be the collection of arcs which are properly contained in some other arc in F. Define $F' = F \backslash C(F)$ to be the proper subfamily of F.

Definition [4]. *For each arc i, define the following functions:*

NEXT(i,F) – *the arc in F whose tail is the first one encountered among the non–overlapping arcs of i in a clockwise traversal from t(i)*

GD(i) – *the maximal independent set of the form i, i_1, ..., i_k, where $i_1 = NEXT(i)$ and $i_t = NEXT(i_{t-1})$, $t = 2, ..., k$ (GD is short for GREEDY)*

Below, we list some properties (described in Hsu [4]) of proper circular–arc families which are useful for our algorithm.

(2.1) $|GD(i)|$ *is maximum among all independent sets containing i.*

(2.2) Let i be any arc in a proper family F. Then $\alpha(F)-1 \leq |GD(i)| \leq \alpha(F)$.

(2.3) Let i be any arc in a proper family F. Then $|GD(i)| \leq |GD(NEXT(i))|$.

Consider the directed graph $D = (V_d, E_d)$ with $V_d = F'$ and $(i,j) \in E_d$ iff $j = NEXT(i)$. Each vertex in D has out–degree 1. Define $NEXT^2(i) = NEXT(NEXT(i))$ and $NEXT^k(i) = NEXT(NEXT^{k-1}(i))$ for $k > 1$. The following theorem provides the key property for Hsu and Tsai's algorithm [5]. The latter algorithm simply searches for the first arc u that is contained in a directed cycle of D and computes GD(u).

Theorem 2.4 [4]. *Let C be a directed cycles in D. Then* $|GD(u)| = \alpha(F)$ *for any u in C.*

3. ANALYSIS OF OPTIMAL TWO–INDEPENDENT SET PROBLEMS

Denote a pair of disjoint independent sets S_0 and S_1 in an arc family F by $(\mathbf{S_0,S_1})$. (S_0,S_1) is said to be an ***optimal pair*** in F if $|S_0 \cup S_1|$ is maximum. We shall describe some properties of optimal pairs which leads to the search algorithm in the next section.

Consider the digraph D defined in Section 2. Since each vertex has out–degree 1, there exists at least one directed cycle in D and no two directed cycles can share any vertex. Hence, each component of D contains a unique directed cycle. We say D is ***streamlined*** if it has a unique cycle C (namely, D is connected) and $|C|$ equals $\alpha(F)$. If D is not streamlined, then the following algorithm easily finds an optimal pair: Find one maximum independent set S in F first, find a maximum independent set S' in F\S, and report (S,S').

From now on, we shall assume that F has a streamlined digraph D which contains a unique cycle C of size $\alpha(F)$, and $F \neq C$. The following theorem greatly simplifies our algorithm.

Theorem 3.1 *There exists an optimal pair* (S_0,S_1) *such that* $C \subseteq S_0 \cup S_1$.

By theorem 3.1, it suffices to find a subset B in F\C such that

(a) $B \cup C$ is the union of two disjoint independent sets S_0 and S_1 in F; and

(b) (S_0,S_1) is an optimal pair.

We call a subset B satisfying (a) a ***feasible*** subset in F. A feasible subset does not have to be an independent set itself. Let F' be the proper subfamily of F\C. It can be shown that our search for an optimal feasible set can be confined to F'.

Define $\beta(F')$ to be the size of a maximum feasible subset in F'. We have that $\alpha_2(F) = \alpha(F) + \beta(F')$. Hence, in the remainder of this paper, we shall concentrate on determining $\beta(F')$. Let X be a subset of F', define $\beta(X,F')$ to be the size of a maximum feasible set in F' containing X. Denote $\beta(\{i\},F')$ by $\beta(i)$. Clearly, there exists some arc i such that $\beta(i) = \beta(F')$. Similar to the maximum independent set algorithm in [8], we shall adopt a greedy method for determining $\beta(F')$.

Let i be an arc in F'. Our approach for finding $\beta(i)$ is to construct a greedy feasible set starting with i by defining an appropriate NEXT function in F'. This construction is complicated by the fact that, not only we need know which arc in F' should be chosen as the greedy arc after i, we also have to decide which subset this arc is assigned to.

Definition. *For any endpoint p, define a linear order "$<_p$" as the clockwise order starting from p as follows: let* i_1, i_2 *be any two arcs whose tails are different from p. Then* $i_1 <_p i_2$ *iff* $t(i_1) \in (p,t(i_2))$. *If p equals t(i) for some arc i, denote* $<_p$ *by* $<_i$.

Definition. *For any arc i in F', define two types of greedy arcs, denoted by* $NB_1(i)$ *and* $NB_2(i)$, *and their subset change indicator functions* $T_1(i)$, $T_2(i)$ ($T_1(i) = 0$ *if* $NB_1(i)$ *is assigned to the same subset as i, and* $T_1(i) = 1$, *otherwise; similarly for* $T_2(i)$) *as follows.*

(1) *If* $NEXT(i) \notin C$, *then* $NB_1(i) = NB_2(i) = NEXT(i)$, *and* $T_1(i) = T_2(i) = 0$.

(2) *If* $NEXT(i) \in C$, *then let* $NEXT(i) = c_m$ *and define the greedy arc and its assignment according to the following two cases (shown in Figure 3.1):*

(a) $c_m >_i c_{m+1}$: *then* $NB_1(i) = NEXT(c_m,F')$, $T_1(i) = 1$ *and* $NB_2(i) = NEXT(i,F')$, $T_2(i) = 0$.

(b) $c_m <_i c_{m+1}$: *then* $NB_1(i) = NEXT(i,F')$, $T_1(i) = 0$ *and* $NB_2(i) = NEXT(c_m,F')$, $T_2(i) = 1$.

Lemma 3.2. *For each i in F', there exists a feasible set* $B = \{i(=i_1),i_2,\ldots,i_k\}$ *such* *that $k = \beta(i)$ and each i_{m+1}, for $m = 1, .., k-1$, is either $NB_1(i_m)$ or $NB_2(i_m)$.*

Although Lemma 3.2 provides a method for finding an optimal feasible set in F' using $NB_1(\cdot)$ and $NB_2(\cdot)$, the number of possible combinations of $NB_1(i_m)$ and $NB_2(i_m)$ is still astronomical. In the remainder of this section, we shall show that, in fact, it is sufficient to apply "only" $NB_1(\cdot)$ on some arc to obtain an optimal feasible set.

For each i in F', define **$GD_1(i)$** to be the maximal feasible set $\{i, NB_1(i), \ldots, NB_1^{k-1}(i)\}$ such that either $NB_1^{k}(i) = i$ or $GD_1(i) \cup \{NB_1^{k}(i)\}$ is not feasible. Define ***$LAST_1(i)$*** to be $NB_1^{k}(i)$, and **$\beta_1(i)$** to be $|GD_1(i)|$. Let c_1 be the most counterclockwise arc in C that overlaps with i. Define $H_0 = \min_i (h(i), h(c_1))$ and $H_1 = \max_i(h(i), h(c_1))$.

The following algorithm determines $GD_1(i)$ in $O(n)$ time.

ALGORITHM 2. Find $GD_1(i)$ for a specified arc i in F':

```
1.  Let $S_{t^*}$ be the set i is assigned to initially, where $t^* = 0$ if $H_0 = h(i)$ and
    $t^* = 1$ if $H_0 = h(c_1)$.
2.  Let $GD_1(i) = \{i\}$, $j = NB_1(i)$, and $t = t^* + T_1(i) \pmod 2$;
3.  While $t(j) <_{c_1} H_t$ do
          $GD_1(i) = GD_1(i) \cup \{j\}$;    /* assign j to $S_t$ */
          $j = NB_1(j)$; $t = t + T_1(j) \pmod 2$;
```

Lemma 3.3. *Suppose F is an interval family in which no arc passes through point p. Let j be the arc with the left–most right endpoint in F'. Then $\beta_1(j) = \beta(j) = \beta(F')$.*

Theorem 3.4. *Let F be a circular–arc family with a streamlined digraph. There exists an arc i in F' such that $\beta_1(i) = \beta(F')$.*

4. ALGORITHM FOR MTIS PROBLEM ON CIRCULAR–ARC GRAPHS

Again, assume that F is an arc family with a streamlined directed graph D which contains a unique cycle C of size $\alpha(F)$, and $F \neq C$. In [8], Hsu and Tsai proved that for any arc i, j, the difference between GD(i) and GD(j) is no more than one. Similar to this, we can prove that the difference between $GD_1(i)$ and $GD_1(j)$ is no more than two. This is an important fact used in our algorithm.

Theorem 4.1 $|\beta_1(i) - \beta_1(j)| \le 2$.

In order to prove theorem 4.1, we need the following three lemmas. The first lemma introduce the "non–decreasing" property of $NB_1(\cdot)$:

Lemma 4.2. *Let i and j be any two arcs in F', $i' = NB_1(i)$, and $j' = NB_1(j)$. If $i' \neq j'$, then segment $(t(i), t(i'))$ does not contain segment $(t(j), t(j'))$.*

Lemma 4.3. *Let i be any arc in F', $GD_1(i) = \{i_1(=i), i_2, i_3, \ldots, i_p\}$. Let $i_{p+1} = NB_1(i_p)$, $i_{p+2} = NB_1(i_{p+1})$, and $i_{p+3} = NB_1(i_{p+2})$. Let H_0 and H_1 be those defined in* *the initial assignment. Denote $<_i$ by $<$. We have*

(1) *The tail of every arc in $GD_1(i)$ is contained in segment $(t(i), H_1)$.*

(2) *The arcs in $GD_1(i)$ have increasing right endpoints, namely, $i_2 < \ldots < i_{p-1} < i_p$.*

(3) *If $i_{p+1} \neq i$ and $i_{p+2} \neq i$, then $t(i)$ is either in segment $(t(i_{p+1}),t(i_{p+2}))$ or in segment $(t(i_{p+2}),t(i_{p+3}))$.*

Lemma 4.4. *Let i, j be any two arcs in F', and $GD_1(i) = \{i_1(=i), i_2, i_3, \ldots ,i_p\}$. If $NB_1(j) \neq i$, then segment $(t(i_p), t(i))$ is not contained in segment $(t(j), t(NB_1(j)))$.*

To calculate $\beta(F')$, one has to calculate $\beta_1(i)$ for every i in F'. If we apply algorithm 2 repeatedly to each i in F', it would take $O(n^2)$ time. Instead, we shall apply depth–first–search to construct a linear time algorithm. Note that if $\beta_1(i) = k$, for some i in F', then $k-2 \leq \beta_1(j) \leq k+2$ for any j in F' by Theorem 4.1. Hence, it suffices to check at most five elements $NB_1^m(j)$, where $m = k-2,\ldots,k+2$, to determine $\beta_1(j)$. In the following we describe how to determine $NB_1^k(j)$ for any fixed k efficiently.

Consider the digraph $D_1 = (F',E_1)$, where $(i,j) \in E_1$ iff $j = NB_1(i)$. Since every vertex in D_1 has out–degree one, there is a unique directed cycle in each component of D_1 and $|F'| = |E|$. If D_1 is not connected, then we can deal with each component separately. Hence, assume it is connected. There exists exactly one cycle, say $C^* = \{j_1,\ldots,j_p\}$, in D_1. Every vertex in C^* is said to be a ***kernel vertex***. Initially, we determine $\beta(j_1)$ using ALGORITHM 2. For every other kernel vertex j_m we have $NB_1^k(j_m) = j_{m+k-1(\text{mod } p)}$, which takes constant time.

For any other vertex j, $NB^k(j)$ can be determined as follows. If we delete all edges in C^*, the remaining graph becomes a forest with each tree containing exactly one kernel vertex (called its **root**). Now, consider a specific tree with root vertex j_m. Define a depth function on this tree as follows. Let the depth of each kernel vertex be 0. For every other vertex i, depth(i) is defined to be the length of the shortest path from i to its root vertex.

Apply DFS (depth first search) to the tree starting with j_m. Whenever a vertex j is pushed on top of the stack, the elements in the stack give the unique path from j to the root vertex j_m. If $\text{depth}(j) \leq k$, then $NB_1^{\text{depth}(j)}(j)$ is j_m, and $NB_1^k(j) = j_{m+k-\text{depth}(j)(\text{mod } p)}$, which can be determined in constant time. On the other hand, if $\text{depth}(j) > k$, then $\text{depth}(NB_1^k(j)) = \text{depth}(j) - k$. In the latter case, $NB_1^k(j)$ can be identified by maintaining a pointer which always points at the k–th element below the top element j. Since every vertex is pushed and popped exactly once in DFS, the time complexity of our modified DFS is still $O(n)$.

ALGORITHM 3. Calculating $\beta(F')$ for the case that D is streamlined

```
1.  Arbitrarily select an arc i in F'. Let $m = \beta_1(i) - 2$.
2.  Calculate $NB_1^m(j)$ and $\sum_{i=1}^{m} T_1^i(j)$, for every arc j in F'.
3.  For every arc j, find $\beta_1(j)$.
4.  Let $\beta(F') = \max_j \beta_1(j)$.
```

Using the above argument, $\sum_{i=1}^{m} T_1^i(j)$ for all the j in F' can be calculated in O(n) too. Therefore, step 2 takes O(n). Since $\beta_1(j) \leq m+4$, for each j in F', the calculation of $\beta_1(j)$ in step 3 takes constant time. Hence, step 3 takes O(n). The time complexity of algorithm 3 is therefore O(n).

5. THE TWO TRACK ASSIGNMENT (TTA) PROBLEM

In Section 4, we show that it is sufficient to consider the $NB_1(\cdot)$ function for constructing optimal pairs for the MTIS problem on circular–arc graphs. However, this is no longer the case for the two–track assignment problem. As illustrated by the example in Figure 5.1(a), one has to use $NB_2()$ once to get an optimal solution. Furthermore, the sequence of $NB_1()$ and $NB_2()$ may vary as illustrated in Figure 5.1(a) and 5.1(b). Hence, a greedy approach similar to that used in Section 3 and 4 is not likely to solve the TTA problem directly. Instead, we shall transform the TTA problem into an MTIS problem on circular–arc graphs.

Consider a family H of intervals and two interval tracks (a_1,b_1) and (a_2,b_2). Without loss of generality, assume $a_1 < a_2$. Let $b_{max} = \max(b_1,b_2)$, $b_{min} = \min(b_1,b_2)$. Denote by (R_1,R_2) a feasible pair of independent sets for the TTA problem, where R_j is an independent set of intervals in H assigned to track (a_j,b_j), j = 1, 2. Without loss of generality, assume all intervals in H are contained in segment (a_1,b_{max}). Let p and p' be two points such that $p < a_1$ and $p' > b_{max}$. By identifying p with p', H is transformed into a family of circular arcs. Add two new arcs $i_1 = (b_1,a_1)$ and $i_2 = (b_2,a_2)$ and let $F = H \cup \{i_1,i_2\}$. We have the following

Lemma 5.1. If (R_1,R_2) is a feasible two–track assignment on H, then $(R_1 \cup \{i_1\}, R_2 \cup \{i_2\})$ is a feasible two–independent sets in F.

Corollary 5.2. If there exists an optimal pair (S_1,S_2) for the MTIS problem on F such that $i_1 \in S_1$ and $i_2 \in S_2$, then $(S_1\backslash\{i_1\},S_2\backslash\{i_2\})$ is an optimal pair for the TTA problem on H.

Unfortunately, such an optimal pair might not exist in F. Below, we describe a condition that guarantees the existence of such an optimal pair. We then show that one can always modify H to satisfy this condition.

(5.3) *There is at most one interval in H whose head is less than a_2 and at most one* *interval whose tail is greater than* $b_{\min}$.

Lemma 5.5. *Suppose H satisfies (5.3). Then there exists an optimal pair* (S_1,S_2) *in F such that* $i_1 \in S_1$ *and* $i_2 \in S_2$.

To describe the modification on H, we need define the following NEXT(·,H) function:

NEXT(i,H) – *the interval in H whose tail is first encountered among all non–overlapping intervals of i in a right traversal from t(i)*

Lemma 5.5. *Any interval family H can be modified to a family* H_1 *satisfying condition (5.3) such that* $\tau(H) = \tau(H_1) + r$, *for some constant r.*

The above transformation from an arbitrary interval family H to one that satisfies (5.3) can be easily carried out in O(n) time. Since the TTA problem on the latter family is equivalent to the MTIS problem on its corresponding arc family, which is solvable in O(n) time, the original TTA problem on H can be solved in O(n) time given the sorted endpoint list.

REFERENCES

1. F. Gavril, *Algorithms on circular arc graphs*, **Networks**, 4 (1974), pp. 357–369.
2. U.I. Gupta, D.T. Lee and J. Y.–T. Leung, *Efficient algorithms for interval graphs and circular–arc graphs*, **Networks**, 12 (1982), pp. 459–467.
3. W.–L. Hsu, *Maximum weight clique algorithm for circular arc graphs and circle graphs*, **SIAM J. Comput.**, 14 (1985), pp. 224–231.
4. W.–L. Hsu, *A simple linear time algorithm for the maximum independent set problem on circular–arc graphs*, (1988), Technical Report, Northwestern University.
5. W.–L. Hsu and K.–H. Tsai, *Linear time algorithms on circular–arc graphs*, (1988), Proc. 26th Allerton Conference on Communication, Control and Computing.
6. D.S. Johnson, *The NP–completeness column: An ongoing guide*, **J. Algorithms**, 6 (1985), pp. 434–451.
7. S. Masuda and K. Nakajima, *An optimal algorithm for finding a maximum independent set of a circular–arc graph*, **SIAM J. Comput.**, 17 (1988), pp. 41–52.
8. M. Sarrafzadeh and D.T. Lee, *A new approach to topological via minimization*, to appear in **IEEE Transactions on Computer–Aided Design.**
9. A.C. Tucker, *An efficient test for circular arc graphs*, **SIAM J. Comput.**, 9 (1980), pp. 1–24.
10. Yannakakis and Gavril, *The maximum k–colorable subgraph problem for*

chordal graphs, **Information Processing Letter**, 24 (1987), pp. 133–137.

11. Yeh, S.S. and A.S. LaPaugh, *Algorithms for finding a maximum bipartite subgraph for special classes of graphs*, Technical Report, Princeton University.

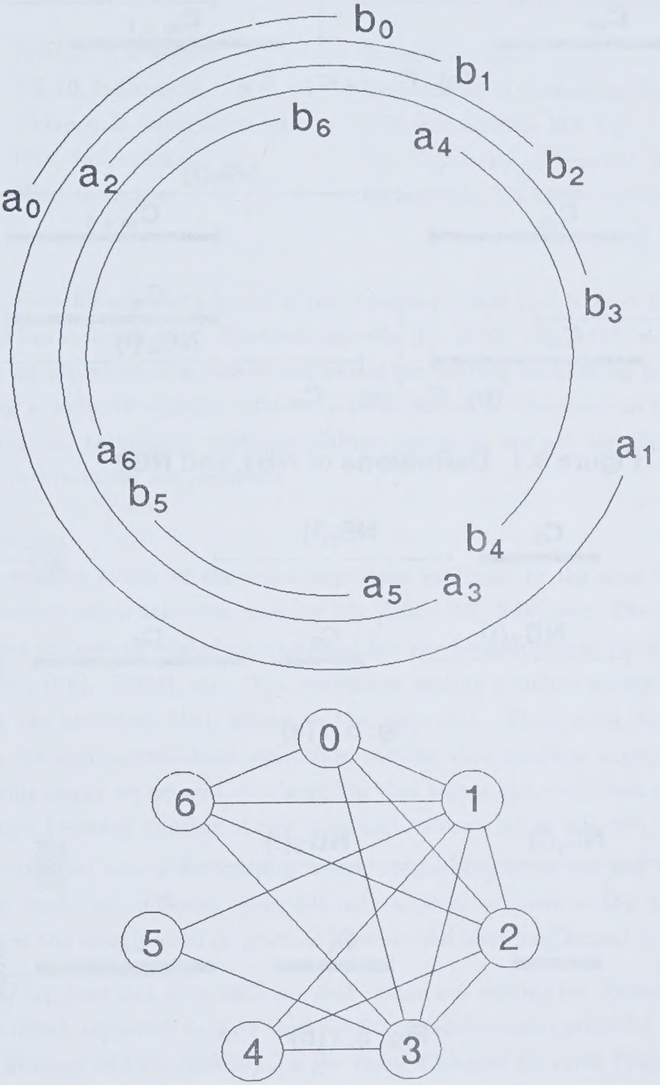
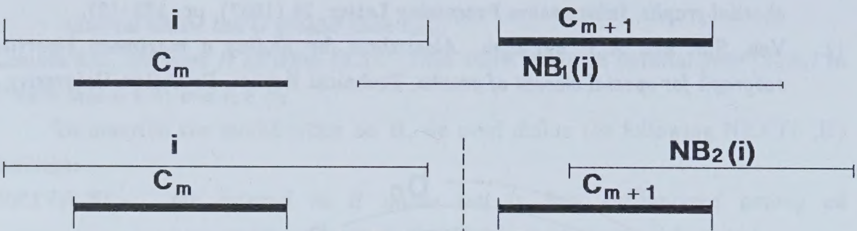
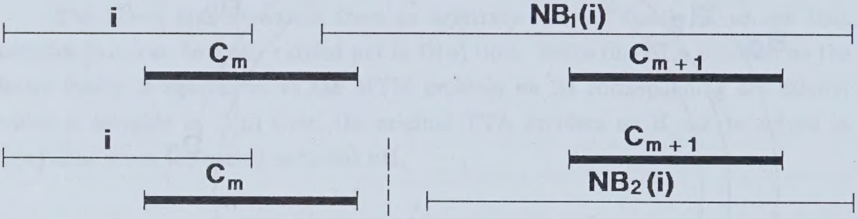

b_0
b_1
b_6
a_4
b_2
a_2
a_0
b_3
a_6
a_1
b_5
b_4
a_5
a_3
0
6
1
5
2
4
3

Figure 1.1. A circular – arc family F


i
C_m
C_{m+1}
$NB_1(i)$
i
C_m
$NB_2(i)$
C_{m+1}

(a) $C_{m+1} <_i C_m$


i
C_m
$NB_1(i)$
C_{m+1}
i
C_m
C_{m+1}
$NB_2(i)$

(b) $C_m <_i C_{m+1}$

Figure 3.1 Definitions of NB1 and NB2

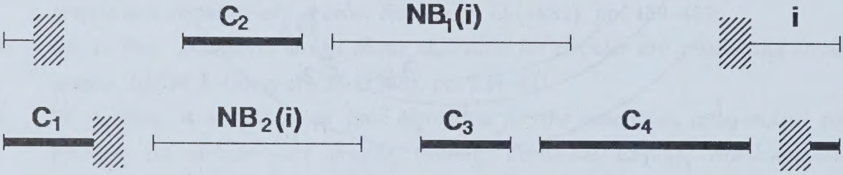

C_2
$NB_1(i)$
i
C_1
$NB_2(i)$
C_3
C_4

Fig. 5.1(a)

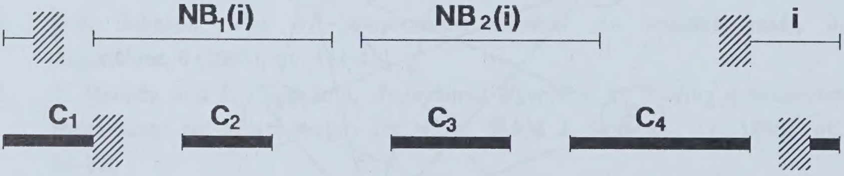

$NB_1(i)$
$NB_2(i)$
i
C_1
C_2
C_3
C_4

Fig. 5.1(b)

Figure 5.1 Optimal sequences for two – track assignment problem

SWITCHBOX ROUTING FOR MULTITERMINAL NETS IN MANHATTAN MODE

Michael Kaufmann
FB 10, Informatik
Universität des Saarlandes
6600 Saarbrücken
West Germany

Ioannis G. Tollis
Department of Computer Science
P.O. Box 830688, MP 3.1
The University of Texas at Dallas
Richardson, TX 75083 - 0688

Abstract. Switchbox routing is one of the most important problems in the area of VLSI layout, and has received much attention recently [F], [KM], [M], [MP], etc. In this paper we present an algorithm that solves any switchbox routing problem in Manhattan mode by inserting a bounded number of new rows and columns. Contrary to the cases known before, using our technique, multiterminal-net problems are not significantly harder to solve than two-terminal net problems.

Motivation.

Switchbox routing is one of the most important problems in the area of VLSI layout, and has received much attention recently [F], [KM], [M], [MP], etc. The channel routing problem is an interesting related problem that has also been extensively investigated [BBL], [GK], [MPS], [PL], [RBM], etc. The switchbox routing problem consists of electrically connecting the *terminals* that belong to the same *net*. There exist no provably good algorithms for multiterminal-net switchbox routing that produce a good layout of the wires. In this paper we present an algorithm that solves any switchbox routing problem by inserting a bounded number of new rows and columns using only two layers.

There are two modes for routing where overlapping wires are not allowed: In the Manhattan mode two different wires are only allowed to cross at the same grid point. This mode is the mostly used in practice because the layer assignment is immediate: All

---

Part of the work was done while the first author was visiting the University of Texas at Dallas. Research supported in part by the DFG, Sonderforschungsbereich 124, Teilproject B2 (VLSI Entwurf und Parallelität) and the Texas Advanced Research Program under grant No. 3972.

horizontal wire segments are placed in the first layer, and all vertical wire segments are placed in the second layer. Vertical and horizontal pieces of the same wire are connected by vias. This wiring requires two layers.

The configuration where two wires bend at the same point is calleda ***knock-knee***. In the knock-knee mode both crossings and knock-knees are allowed. In this mode, the wires can be viewed as paths or trees on the grid, which are pairwise edge-disjoint. The switchbox routing problem where each net has exactly two terminals has been studied extensively. Frank [F] presented necessary and sufficient conditions for the feasibility of any two-terminal switchbox routing problem. Efficient algorithms to construct a solution were presented in [MP], [KM]. Although layouts in knock-knee mode are more efficient, the layer assignment is much more complicated. Lipski [L] proved that it is NP-complete to decide whether a given knock-knee layout is 3-layer wirable or not. Brady and Brown [BB] and Tollis [T] presented algorithms that perform the layer assignment for any given layout using at most four layers. Tollis and Vaguine [TV] presented improved techniques for this layer assignment problem.

However, if one allows multiterminal nets then the problem becomes very difficult. Mehlhorn and Preparata [MP] presented a technique that transforms any multiterminal-net problem into a two-terminal net problem by doubling the number of rows and columns, i.e., the area of the new switchbox is four times the area of the original one. Subsequently, the layer assignment techniques discussed above may require up to four layers in order to properly wire the layout. If one desires a two-layer layout, the straightforward technique requires that the area of the switchbox be again doubled. This results in a switchbox with area eight times the area of the original one.

In the Manhattan mode there are several attempts to get the problem theoretically under control, but without any real success. On the practical side there are many heuristics which work well on real problems, but it is not possible to prove anything about their performance [M]. If we are allowed to insert some new columns/rows, then for channel routing it is easy to find a layout in Manhattan mode by doubling the number of columns. Finally, using a simple left edge algorithm, we obtain a solution with number of rows equal to the density of the problem. For switchboxes, nothing provable is known for Manhattan mode.

We investigate the problem of routing in a $c \times r$-switchbox with multiterminal nets and present a technique that produces layouts in Manhattan mode (two-layer wirable) by inserting no more than $c + r$ new tracks, where a track is a row or a column. We first show how to solve a two-terminal net problem. Next, we generalize our technique to multiterminal net problems. Thus we show that contrary to the cases known before, using our technique, multiterminal net problems are not significantly harder to solve than two-terminal net problems.

Definitions.

A *switchbox* routing problem is given by the routing region as a square grid bounded from the outside by a rectangle. The vertices on the boundary are the *terminals* of the nets. Each *net* consists of at least 2 terminals which need to be electrically connected. If each net has exactly two terminals, we have a two-terminal problem, otherwise a multiterminal problem.

The switchbox consists of several tracks (rows and columns). Each track contains at most two terminals. A cut is a line between two consecutive parallel tracks. Its capacity is given by the number of grid edges crossed by the line, its density is denoted by the number of nets having terminals on opposite sides of the line.

We now present the algorithm for the two-terminal case. It turns out that the generalization to multiterminal nets is straightforward except for one case.

The two-terminal case.

The layouts produced by our algorithms will be simple and nice. Each net will be routed using the minimum number of bends, which of course implies minimum number of vias. We divide the nets into two categories: one-bend nets and two-bend nets. Among the two-bend nets we distinguish one-sided-two-bend nets and two-sided-two-bend nets, see Figure 1.

A *cycle* of nets consists of a sequence $N_1, N_2, ..N_{l+1}$ of 2-sided-2-bend nets where the nets N_i and N_{i+1} share a track for one of their terminals, for $1 \leq i \leq l$, $N_1 = N_{l+1}$ and the other nets are pairwise different. We say that this cycle has length l.

During the algorithm, we will insert a number of new tracks thus creating an *extended switchbox.* Some tracks will be inserted between existing tracks and some next to the boundary. The *center* of the extended switchbox contains exactly the original tracks and the new tracks inserted between them, but not the tracks inserted next to the boundary. The main idea is to insert new tracks such that the center of the layout contains only necessary crossings, see Figure 2.

In the correctness proof each new track is associated with an original track such that after the completion of the algorithm, each original track has at most one associated track.

Theorem 1: *Let P be a two-terminal $r \times c$-switchbox routing problem such that for every cut the density is not greater than the capacity. P can be solved in the Manhattan mode with the minimum number of bends (vias) by inserting at most $c + r$ new tracks.*

Our algorithm is as follows:

1. Route the 1-bend nets in the natural way. Whenever an overlap in track t arises, it is resolved by inserting a new parallel track t' next to t. The 1-bend net which has a terminal in this track, is routed partially in t'. The new track is associated with the original track, where the overlap happened.

2. Route the 1-sided-nets and the 2-sided-2-bend nets which do not belong to a cycle inserting extra tracks close to the boundary for the middle segments. These new tracks are not within the center. They are associated with one of the tracks where the terminals of the nets lie.

3. Route the 2-sided-2-bend nets that belong to cycles as follows: All nets with non-overlapping middle segments are routed as in Step 2. The remaining nets N have to use the first track in the center which is completely free in the interval of the middle segment of N or which has no associated track, so that we can insert a new track.

An example of a layout produced by our algorithm is shown in Figure 3.

Lemma 1: In Step 1 of the algorithm, each track is associated with at most one new track.

Proof: This is obvious by construction. In Step 1, we consider only 1-bend nets. They consist only of 2 segments. And each segment starts at a terminal. Therefore each segment can overlap at most one other segment, such that each track contains only at most one overlap. Each overlap creates a new track and this track is associated to the original track, where the overlap was. ∎

Lemma 2: In the first two steps, each track is associated with at most one new track.

Proof: We have shown in Lemma 1, that the obvious strategy works for Step 1. In Step 2, we have to be more careful. First we route the 1-sided-2-bend nets. Each of them consists of three segments, where the position of the middle segment determines the total layout of the net. We create a new track for the middle segment as close to the terminals as possible, see Figure 4.

This way we avoid possible overlaps with other nets, which have their terminals on the same tracks. Hence, we ensure that we do not need to associate parallel tracks to these tracks. We can associate the new track containing the middle segment with one of the terminal tracks of the net. This depends on the type of net that has its terminal on the same track. If one of them is a 1-bend net, then we can associate this track with the new track, otherwise we can choose one of them, but we have to propagate this choice to the other net.

Note that there are enough tracks such that propagation is always possible. Furthermore, if such a propagation is necessary, we have created a track, which is completely free in the center. We will use it while routing 2-sided-2-bend nets.

After that, we route the 2-sided-2-bend nets, which do not belong to a cycle. We do it in the same way as for the 1-side nets. Note that we have to insert the tracks between the new tracks from the first part of Step 2 and the center to avoid possible overlaps of 1-sided nets and 2-sided-2-bend nets that have terminals on the same track. ∎

Lemma 3: Let l be the number of new tracks inserted during Steps 1 and 2. The routing of the nets that belong to cycles requires the insertion of at most $c + r - l$ new tracks.

Sketch of Proof: We will only consider cycles of length 2. Cycles of larger length can be handled analogously. We route the cycles one by one. Let us consider a cycle of length two with terminals on tracks t_u and t_o, which are w.l.o.g. horizontal. Note that one of these tracks must contain an overlap of the two nets, N_1 and N_2. One of the two nets, N_1, is layed out as described before in the non-cycle case. For the middle segment of N_1, we use an extra vertical track t_x close to the boundary. Track t_x is associated with the terminal track that does not contain any overlap. This is possible since this track does not have an associated track yet. Now we have to find a track for the middle segment of the second net, N_2. Both tracks t_u and t_o have already associated tracks. Hence, the idea is to find a track for the middle segment within the center of the switchbox, that is either completely free or does not have an associated track. In either case that would be an appropriate place for the middle segment. This is established by the following claim whose proof is omitted from this extended abstract:

Claim: For each remaining middle segment of the cycle there exists an appropriate place within the center of the switchbox. ∎

Theorem 1 follows from the three lemmata.

In the next section we show how to extend this technique to routing in a switchbox with multiterminal nets. It turns out that most situations are much easier to do in the multiterminal case.

The multiterminal case.

We extend the above algorithm to the multiterminal case in a quite easy and straightforward way. The main idea is to decompose the terminals of any net into groups. Each group contains all the terminals of a net that lie on the same side of the boundary. Hence, we have at least one and at most four groups of terminals for each net. If a group contains more than one terminal (called a large group), it will be routed like a one-sided net (see Step 2 of the algorithm for the two-terminal case). Large groups on adjacent sides are connected using the new tracks next to the boundary. Since we want to hold the cut condition within the center, we have to ensure that the connections between the different groups will not overlap. Hence, if we have groups lying on opposite sides and at least one of them is large, then the connection between the two groups is established without increasing the density within the center of the layout by simply using the tracks of the terminals.

Every case of multiterminal net, except for one, turns out to be easy or shown before in the two-terminal case, see Figure 5.

The only complicated case happens when a three-terminal net has two terminals on one side and the other terminal on an adjacent side. In this case, we create a track t_u close to the boundary to connect the terminal in the large group. Track t_u will be associated either with t_r or t_l, see Figure 6. A careful case analysis reveals that there is enough space on the existing or associated tracks to accomodate this solution.

A careful runtime analysis shows that our algorithm takes time $O(n \log n)$, where n is the number of tracks of the original switchbox.

Theorem 2: *Let P be a multiterminal $r \times c$-switchbox routing problem such that for every cut the density is not greater than the capacity. P can be solved in the Manhattan mode by inserting at most $c + r$ new tracks. Furthermore, the algorithm that produces the solution takes time $O(n \log n)$.*

Remarks.

If the original switchbox is squarish, the above theorem implies that the area of the new switchbox is at most four times the area of the original one. If it is very tall or wide, much better bounds than given by the straightforward application of the theorem can be derived very easily.

References.

[BBL] B. S. Baker, S. N. Bhatt, and F. T. Leighton: "An Approximation Algorithm for Manhattan Routing," Proc. of the 15th Ann. ACM Symposium on Theory of Computing, pp. 477-486 (1983).

[BB] M. Brady and D. Brown: "VLSI Routing: Four Layers Suffice," MIT VLSI Conference 1984.

[F] A. Frank: "Disjoint Paths in Rectilinear Grids," Combinatorica, Vol. 2. No. 4, 1982, pp. 361-371.

[GK] S. Gao and M. Kaufmann: "Channel Routing of Multiterminal Nets," Proc. of 28th Ann. Symposium on Foundations of Computer Science, pp. 316-325 (1987).

[KM] M. Kaufmann and K. Mehlhorn: "Routing through a Generalized Switchbox," Journal of Algorithms 7, pp. 510-531 (1986).

[L] W. Lipski: "On the Structure of Three-Layer Wirable Layouts," pp. 231-243 in Advances in Computing Research, vol. 2, ed. F.P. Preparata, 1984.

[M] M. Marek-Sadowska: "Two-Dimensional Router for Double Layer Layout," Proc. 22nd Design Automation Conference, 1985, pp. 117-123.

[MP] K. Mehlhorn, F. P. Preparata: "Routing through a Rectangle," Journal of the ACM, Vol 33, pp. 60-85,(1986).

[MPS] K. Mehlhorn, F. P. Preparata, and M. Sarrafzadeh: "Channel Routing in Knock-Knee Mode: Simplified Algorithms and Proofs," Algorithmica 1(2), pp. 213-221, Springer-Verlag (1986).

[PL] F. P. Preparata and W. Lipski: "Optimal Three-Layer Channel Routing," IEEE Trans. on Computers C-33, pp. 427-437 (1984).

[RBM] R. L. Rivest, A. Baratz, and G. Miller: "Provably Good Channel Routing Algorithms," Proc. CMU Conf. on VLSI, pp. 153-159 (Oct. 1981).

[T] I. G. Tollis: "A New Algorithm for Wiring Layouts," pp. 257-267 in VLSI Algorithms and Architectures (Proceedings of AWOC'88), J. Reif (Ed.), LNCS, Vol. 319, Springer-Verlag, (1988)

[TV] I. G. Tollis and A.V. Vaguine: "Improved Techniques for Wiring Layouts in the Square Grid," Proc. 1989 International Symposium on Circuits and Systems, Portland, OR, May 1989, pp. 1875-1878.

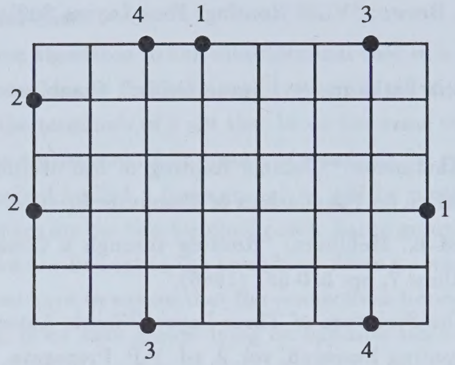


Figure 1. Net 1 is a 1-bend net ; Net 2 is 1-sided-2-bend net; Nets 3 and 4 are 2-sided-2-bend nets and form a cycle.

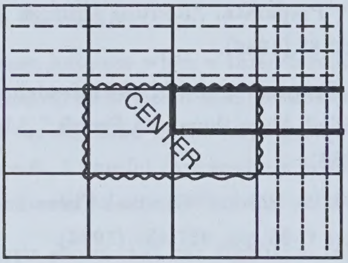


Figure 2. The dashed lines denote new tracks. It is not allowed to route a two-terminal net as shown.

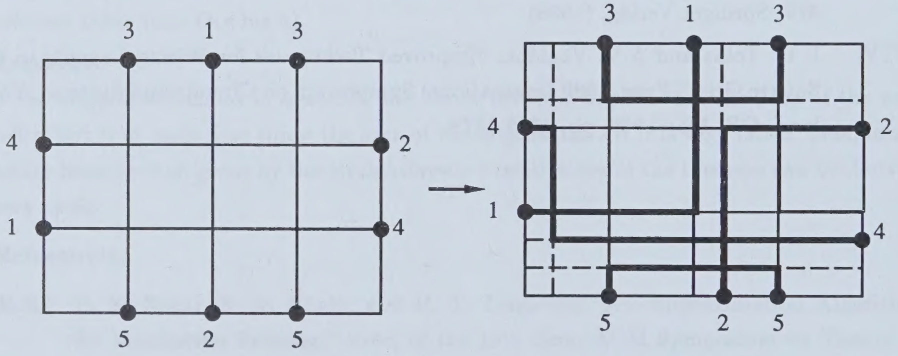

3
1
3
4
2
1
4
5
2
5
3
1
3
4
2
1
4
5
2
5

Figure 3. An example of a layout produced by our algorithm. The dashed lines again denote new tracks.

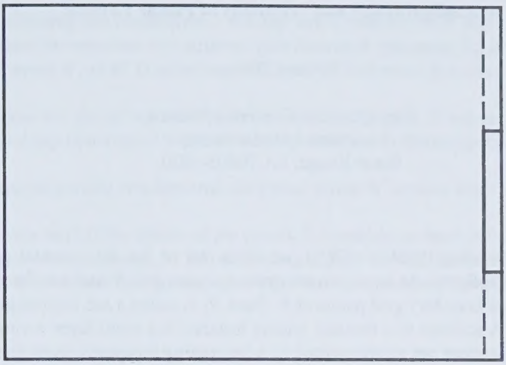


Figure 4. Insertion of a new track for the middle segment of a net.

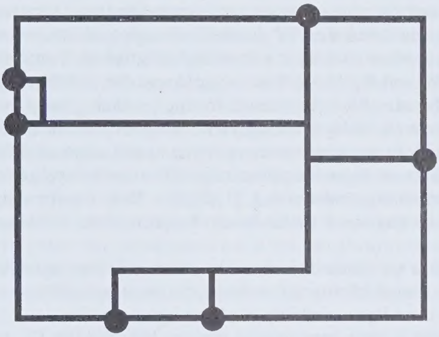


Figure 5. A layout of a multiterminal net.

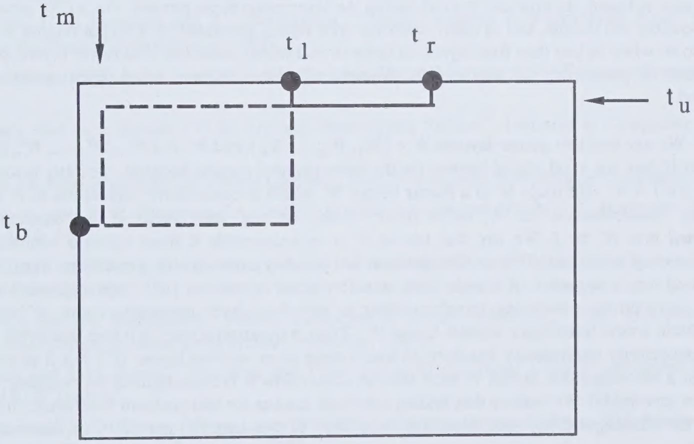


Figure 6. Two alternative ways to connect the leftmost terminal.

Reconfigurability and Wirability of Planar Layouts

Si-Qing Zheng

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020

The Channel Routing Problem (*CRP*), which is one of the fundamental problems in VLSI design automation, is defined as follows. As input, we are given a square grid R and a collection $N = \{N_1, N_2, ..., N_n\}$ of mutually disjoint sets of boundary grid points of R. Each N_i is called a *net*, and the grid points in N_i are called the *terminals* of net N_i. A solution to a channel routing instance is a multi-layer wiring in the grid such that all terminals belonging to the same net are connected by wires runing in several given *conducting layers*, each is a copy of the grid R, in the way that wires connecting different nets do not share a grid point in any layer (note that our definition of *CRP* covers the two-shore channel routing problem and the switch-box routing problem). In this paper, we consider the knock-knee mode *CRP*. Since the three-dimensional structure of a solution to the *CRP* can be very complex and there are many optimality criterion (area, wire length, number of vias and the number of layers, etc.) to be considered, the *CRP* is usually decomposed into two subproblems: (a) find a *planar layout* $W = \{W_1, W_2, ..., W_n\}$, where each W_i is a connected subgraph of R such that it connects all terminals of net N_i and any two distinct W_i and W_j do not share any grid segment, and (b) assign wire segments in W to the given conducting layers. This simplifies the channel routing problem greatly by reducing a problem dealing with three-dimensional objects (multi-layer wirings) to a simpler problem of dealing with two-dimensional objects (wire layouts) so that one can concentrate on some optimization criterion, such as area and channel width; once a wire layout is found, layer assignment algorithms can be applied to obtain a multi-layer wiring solution to the given channel routing problem (e.g. [1][2][3]). Thus, transforming a wire layout into a multi-layer wiring had been recognized as one of the fundamental aspects of the VLSI layout problem.

It is well known that any planar layout can be wired with four layers by efficient algorithms [4][5]. However, when the number of available layers is reduced, the situation is different. Although the necessary and sufficient conditions for the three-layer wirability of planar layouts are known [6], the problem of determining whether or not a planar layout is three-layer wirable remains NP-complete [7]. In [8], [9] and [10], the layer assignment method of ensuring three-layer wirability by layout stretching is investigated, and lower bounds and upper bounds for the wiring area are given. By our experience, finding three-layer assignments is difficult even for some restricted classes of layouts. We believe that the difficulties are partially due to the assumption that once a layout is found, its structure is fixed during the layer assignment process. For a *CRP* instance the solution is most possibly not unique, and in many cases the wire layout generated by a layout finding algorithm may not suitable to be wired in less than four layers. In order to find wiring solutions with fewer layers, one may relex the requirement of preserving the topological structure of planar layouts when layer assignment problem is considered.

We say that two planar layouts $W = \{W_1, W_2, ..., W_n\}$ and $W' = \{W'_1, W'_2, ..., W'_n\}$ are *connectivity equivalent* if they are valid planar layouts for the same channel routing instance. For any layout W, we call the mapping $f(W) = W'$ that maps W to a planar layout W' which is connectivity equivalent to W as a *connectivity preserving transformation* for W; or in other words, we say that layout W is *connectivity equivalently transformed* into W' by f. We say that layout W is *reconfigurable* if there exists a connectivity preserving transformation f such that $f(W)$ is not identical to W. Any connectivity preserving transformation can be decomposed into a sequence of simple basic transformation operations [12]. It is important to notice that by applying connectivity preserving transformations to any three-layer unwirable layout W we know, we can always obtain a new three-layer wirable layout W'. Then, a question arises: Is it true that every planar layout W can be connectivity equivalently transformed into a three-layer wirable layout W'? Put it in another way, does there exist a reconfigurable layout W such that no matter how it is reconfigured the resulting layout is always three-layer unwirable? We believe that finding a definite answer for this problem is difficult. In what follows we consider the reconfigurability and three-layer wirability of two-terminal-net *CRP*, in which each net has two terminals.

We say that a layout W in grid R is a *full layout* if every grid segment of R is occupied by a wire segment of W. It is easy to see that if a layout W is not full, one can always add wires in R to obtain a full layout W' in R without affecting the connectivity. We say that a wire W_i of W in R is *monotone* if any of the horizontal and vertical lines between any two adjacent grid lines of R intersects W_i at most once. We say that a layout W is monotone if every W_i of W is monotone. We have the following theorems.

Theorem 1: A two-terminal-net planar layout W in R is non-reconfigurable if and only if W is a monotone full layout without any two distinct wires W_i and W_j sharing two distinct grid points of R.

Theorem 2: Every non-reconfigurable two-terminal-net planar layout W is three-layer wirable.

Readers may refer to [11] for details of the proofs. It should be pointed out that if every wire W_i in W is represented by a list of turning points, in the order they appear on W_i, then both problems of testing the reconfigurability of two-terminal-net layouts and three-layer wiring non-reconfigurable two-terminal-net layouts are solvable in $O(n\log n)$ time.

The concept of connectivity preserving transformation allows us to introduce an additional phase between layout finding and layer assignment, a phase to refine the topology of the layout, according to some optimality criterion, by performing connectivity preserving transformations. There are a few optimality criterion may be considered when we reconfigure a layout: wire length, number of wire bends, number of vias (contact cuts) that may appear in the multi-layer wiring, and the wirability of the layouts. All of these aspects deserve further investigations. A particular routing algorithm utilizing layout transformations may not consist of all three phases. In many situations the operations of reconfiguring layouts can be incorporated into the layer-assignment phase. With regard to the possibility of improving wirability of layouts through connectivity preserving transformations, several special cases of the open problem mentioned earlier may be of theoretical interest. Is it possible to reconfigure any two-terminal-net layout W to obtain W', which is connectivity equivalent to W, such that W' is three-layer wirable? Is it possible to reconfigure any monotone (full) two-terminal-net layout W to obtain W', which is connectivity equivalent to W, such that W' is three-layer wirable? A more realistic problem is to determine the three-layer wirability of a layout of fixed dimensions under the connectivity preserving transformations. The solution to this problem is useful in obtaining three-layer wirings by stretching the layouts [8][9][10]. Under connectivity preserving transformations, if one can always construct a three-layer wiring for a x-row by y-column layout with large x and y, it is possible to obtain a wiring with smaller additional area. The approach of improving layout quality by connectivity preserving transformations can also be extended to other wiring models.

Reference

[1] K. Mehlhorn and F. P. Preparata, "Routing Through a Rectangle", *J. ACM*, vol. 33, no. 1, 1986.

[2] K. Mehlhorn, F. P. Preparata and M. Sarrafzadeh, "Channel Routing in Knock-knee Mode: Simplified Algorithms and Proofs", *Algorithmica*, 1986.

[3] F. P. Preparata and W. Lipski, Jr., "Optimal Three-layer Channel Routing", *IEEE Trans. Computers*, 33, 5, 1984.

[4] M. L. Brady and D. J. Brown, "VLSI Routing: Four Layers Suffice", *Advances in Computing Research*, vol.2, 1984.

[5] I. G. Tollis, "A New Algorithm for Wiring Layouts", *Proc. 3rd Aegean Workshop on Computing*, 1988.

[6] W. Lipski, Jr., and F. P. Preparata, "A Unified Approach to Layout Wirability", *Mathematical Systems Theory*, vol. 19, 1987.

[7] W. Lipski, Jr., "An NP-complete Problem Related to Three-layer Channel Routing", *Advances in Computing Research*, vol.2, 1984.

[8] M. L. Brady and M. Sarrafzadeh, "Layout Stretching to Ensure Wirability," *Proc. 21st Johns Hopkin Conference on Communication, Control and Computing*, 1987.

[9] T. Gonzalez and S.-Q. Zheng, "Stretching and Three-Layer Wiring Planar Layouts", to appear *INTEGRATION: the VLSI Journal.*

[10] T. Gonzalez and S.-Q. Zheng, "Three-Layer Wirability of Planar Layouts", *Proc. 25th Allerton Conf. on Communication, Control, and Computing*, 1987.

[11] S.-Q. Zheng, "Reconfigurability and Wirability of VLSI Layouts", Technical Report #88-027, LSU, 1988.

[12] S.-Q. Zheng and W. Zhu, "Reducing Wire Length of VLSI Layouts by Connectivity Preserving Transformations", Technical Report #89-007, LSU, 1989.

ON THE PERFORMANCE OF MULTIDIMENSIONAL PHASE MODULATED TRELLIS CODES[1]

LANCE C. PEREZ and DANIEL J. COSTELLO, JR.
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

1 Introduction

Demands for higher information rates (spectral efficiencies) in satellite communications are such that this traditionally power limited channel is now considered to be bandlimited as well. This has made high speed satellite channels a natural candidate for the application of Trellis Coded Modulation (TCM). Introduced by Ungerboeck [1], TCM achieves significant asymptotic coding gains relative to an uncoded system of the same information rate with no bandwidth expansion. Due to the extensive use of nonlinear amplifiers, such as traveling wave tubes, in satellite communications systems only TCM schemes using MPSK signal constellations are appropriate. A class of multidimensional (multi-D) MPSK trellis codes has been proposed for use in NASA high speed satellite communication links [2]. These codes have greater flexibility in achievable information rate, better rotational invariance properties, and higher speed decoder implementations than regular two dimensional (2-D) Ungerboeck codes.

In this paper, computer simulation is used to study the performance of multi-D MPSK trellis codes on the Additive White Gaussian Noise (AWGN) channel and their application to concatenated coding systems. Due to the nonlinearity of trellis codes a number of nonzero information sequences are considered in each simulation and the code performance is determined as the average over these sequences. The bit error rate (BER) of a number of codes is obtained and compared to similar Ungerboeck codes and performance bounds where appropriate. The performance of these codes as inner codes in a concatenated coding system using Reed-Solomon (RS) outer codes over $GF(2^b)$ is also studied. Computer simulation is used to determine the symbol error rate out of the inner decoder where the symbol size of the RS code, b, is chosen to be an integer multiple of the number of bits on a branch of the inner code.

---

[1] This work was supported by NASA Grants NAG5-557 and NGT-70109 and by NSF Grant NCR89-03429

2 Multi-Dimensional Trellis Codes

The fundamental concept of 2D TCM is that by trellis coding onto an expanded signal set relative to that needed for uncoded transmission, a larger free distance and an associated improvement in performance may be achieved without bandwidth expansion or a reduction in information rate. In multi-D MPSK TCM, the same principal is applied to a signal constellation, denoted LxMPSK, constructed from the Cartesian product of L 2D-MPSK signal sets. Each branch of the trellis is then labelled with an ordered sequence of L 2D signals and a 2L-dimensional signal is realized by transmitting a sequence of L 2D signals. The efficient labelling of trellis branches is accomplished through a systematic binary set partitioning technique based on block codes and their cosets.

The class of LxMPSK codes studied here has the same schematic encoder diagram introduced by Forney et.al.[3] for Ungerboeck codes with one slight modification. As shown in Figure 1, the multi-D encoder allows the q least significant bits to be fixed, nominally to 0, during each coding interval. Thus, whereas 2D-MPSK TCM allowed only rate $m/m+1$ codes with information rates (R) of 1.0, 2.0, and 3.0 bits per modulation interval(T) for QPSK, 8PSK, and 16PSK, respectively, multi-D MP3K allows the construction of rate $(L\mathrm{x}\log_2(M-q-1))/(L\mathrm{x}\log_2 M)$ codes with a variety of information rates ranging from 1.0 bit/T to 3.75 bits/T.

As an example, consider the familiar 4-state Ungerboeck 8PSK code with information rate 2.0 bits/T. The fundamental trellis structure of this code is determined by the rate 1/2 convolutional encoder and is shown in Figure 2. During each coding interval, the trellis encoder receives $m = 2$ information bits, of which $\tilde{m} = 1$ are encoded by the rate 1/2 convolutional encoder, and maps them to a single 8PSK signal. This results in the trellis structure of Figure 2 with each transition replaced by $2^{m-\tilde{m}} = 2^1$ parallel transitions. The same rate 1/2 convolutional encoder can be used to construct a 2x8PSK code with information rate 2.5 bits/T. In this case, the trellis encoder receives $m = 5$ information bits per coding interval, of which the least significant bit is encoded by the convolutional encoder. The $m + 1 = 6$ bits are then mapped to a single 2x8PSK signal. This results in a rate 5/6 code with the trellis structure of Figure 2 where each branch is replaced with $2^{m-\tilde{m}} = 2^4 = 16$ parallel transitions and each transition is labelled with a 2x8PSK signal. If the first $q = 1$ bits into the trellis encoder are now set to 0, a rate 4/6, 2x8PSK code with information rate 2 bits/T is obtained. This code has the trellis structure of Figure 2 with $2^{4-1} = 8$ parallel transitions and each transition is labelled with a 2x8PSK signal from the left branch of the first level of the 2x8PSK binary partition (see Pietrobon [2] et. al. for a discussion of multi-D signal set partitioning). It should be noted that, though this example illustrates the concept of these LxMPSK trellis codes, the best LxMPSK codes will in general not have the same convolutional encoder as a 2D Ungerboeck code.

3 Code Performance and Simulation Results

The BER peformance curves for a number of multi-D LxMPSK codes were obtained by computer simulation of the AWGN channel and a soft decision Viterbi decoder. The truncation length of the decoder was chosen to be well in excess of that necessary for asymptotically maximum likelihood performance [4]. Zehavi and Wolf [5] have shown that for a large class of TCM codes a tranfer function upper bound on the average probably of error can be calculated using a modified state diagram approach similar to that for convolutional codes. This procedure becomes impractical as the number of states increases. For multi-dimensional trellis codes, Zehavi and Wolf's method becomes even more intractable due to an exponential increase in the number of error vectors whose distances from the correct path must be enumerated. For a few simple codes, however, simulation results are compared to the Zehavi and Wolf bound and found to conform closely except at low signal to noise ratios (SNR's) where the bound becomes greater than 1. This is of practical interest since many satellite communications systems operate at these SNR's and concatenated coding systems are sensitive to small deviations in the performance of the inner code.

Figure 3 shows the simulation results for a group of $\nu = 2$, Lx8PSK trellis codes with information rate 2 bits/T and $d^2_{free} = 4.0$. These codes show a coding gain from 2.0dB to 2.4dB at $\sim 10^{-5}$ compared to uncoded QPSK. The 4, 6, and 8 dimensional codes suffer a 0.2dB to 0.4dB loss in performance compared to the Ungerboeck 8PSK code due to higher path multiplicities and denser distance spectra. For example, the 4x8PSK code has a multiplicity $N_{free} = 12$ of the minimum free distance path ($d^2_{free} = 4.0$) and a second spectral line at $d^2_2 = 4.686$ with multiplicity $N_2 = 32$. On the other hand, it may be possible to implement a 4x8PSK decoder at 4 times the speed fo the Ungerboeck code by using read only memories (ROM's) to decode the parallel transitions. Analagous simulation results were obtained for the multi-D Lx4PSK codes with information rate 1 bit/T and the Lx16PSK codes with information rate 3 bits/T.

Figure 4 shows the simulation results for a number of codes with information rates not achievable with conventional 2D TCM. This performance is typical of codes with higher rates. This figure also shows the Zehavi and Wolf bound for the 2x8PSK, $R = 2.5$ bits/T, $\nu = 2$ code. Figure 5 shows simulation results for $\nu = 4$ and $\nu = 6$, $R = 2.67$

bits/T, 3x8PSK codes compared to a $\nu = 4$ 8PSK periodically time varying trellis code (PTVTC) of the same effective rate[6]. The 3x8PSK codes show an improvement of 0.5 dB for $\nu = 4$ and 1.0 dB for $\nu = 6$. Additionally, the 3x8PSK codes are capable of a decoder implementation that would operate at three times the speed of the PTVTC code.

4 Concatenated Coding and LxMPSK Codes

Many satellite communications links use concatenated coding in order to achieve extremely low BER's over noisy channels with reasonable code complexity. The class of Reed-Solomon (RS) block codes are often chosen as the outer code in these systems due to their optimal distance properties, high rates, and the availability of relatively simple decoding algorithms. Additionally, an interleaver is usually placed between the inner and outer code to ensure that errors in the RS symbols are independent. For a concatenated coding system using a t-error-correcting (N, K) RS outer code and ideal interleaving, the overall bit error rate is well approximated by

$$p_b = \frac{2t+1}{N} \sum_{i=t+1}^{N} \binom{N}{i} p_s^i (1 - p_s)^{N-i} \tag{1}$$

where p_s is the RS symbol error rate out of the inner decoder. Computer simulation is necessary to get an accurate measure of p_s for a given inner code.

When using convolutional inner codes, it was shown in [7] that choosing an RS code over $GF(2^b)$ matched to a byte oriented inner convolutional codes of rate k/b resulted in an improvement in the overall sytem BER by nearly an order of magnitude compared to unmatched inner codes of the same rate. Costello and Deng[8] have suggested that when using multi-D LxMPSK inner codes and RS outer codes similar results might apply. In Figure 6, the BER and symbol error rate (SER) for a $\nu = 2$, $R_{eff} = 2$ bits/T, 4x8PSK code and the comparable 2D Ungerboeck code are shown for a symbol size of 8 bits, i.e. the RS code is over $GF(2^8)$. The results indicate that the BER penalty of 0.2dB to 0.4dB that multi-D codes suffer compared to 2D codes, as noted above, disappears when SER is considered. Thus, the multi-D codes appear to be better suited for concatenated coding than the 2D codes.

5 Conclusion

The performance of a class of multi-D LxMPSK codes was studied through extensive computer simulation. When compared to 2D Ungerboeck codes with the same information rate, the multi-dimensional codes performed slightly worse due to their high path multiplicities and dense spectra. Additionally, they have a higher trellis complexity due to the larger number of parallel transitions. However, these codes have better rotational invariance properties and are capable of achieving much higher decoder speeds. Furthermore, multi-D LxMPSK codes can be constructed using rates not possible with conventional 2D-MPSK codes. It was also shown that the byte oriented structure of the multi-D codes compensated for the slight loss in BER so that a concatenated system with the proper RS outer code would perform equally as well as a 2D code.

6 References

[1] G. Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets, Parts I and II," *IEEE Commun. Mag.*. Volume 25, Number 2, pp. 5–21, February 1987,

[2] S. S. Pietrobon, R. H. Deng, A. Lafanechere, G. Ungerboeck, and D. J. Costello, Jr., "Trellis Coded Multi-Dimensional Phase Modulation, " to appear in *IEEE Trans. Inform. Theory.*

[3] G. D. Forney, Jr., R. G. Gallager, G. R. Lang, F. M. Longstaff, and S. U. Qureshi, "Efficient Modulation for Band-Limited Channels," *IEEE Jour. S.A.C.*, **SAC-2**, pp. 632–647, September 1984.

[4] J. M. Nowack and M. A. Herro, "The Effect of Viterbi Decoder Path Memory on the Performance of Trellis Coded Modulation " *Proceedings of the 1989 Conference on Information Sciences and Systems*, pp. 735-740, The Johns Hopkins University, Baltimore, Maryland, March 22-24 1989. pp. 12–21, February 1987.

[5] E. Zehavi and J. K. Wolf, "On the Performance Evaluation of Trellis Codes," *IEEE Trans. Inform. Theory*, **IT-33**, pp. 196–202, March 1987.

[6] F. Hemmati and R. J. F. Fang, "Low Complexity Coding Methods for High-Data-Rate Channels," *COMSAT Technical Review*, Volume 16, Number 2, pp. 425–447, Fall 1986.

[7] Lin-nan Lee, "Concatenated Coding Systems Emplying a Unit-Memory Convolutional Code and a Byte-Oriented Decoding Algorithm," *IEEE Trans. Commun.*. **COM-25**, pp. 1064–1074, October 1977.

[8] R. H. Deng and D. J. Costello Jr., "High Rate Concatenated Coding Systems Using Multi-Dimensional Bandwidth Efficient Trellis Inner Codes," to appear in *IEEE Trans. Commun.*.

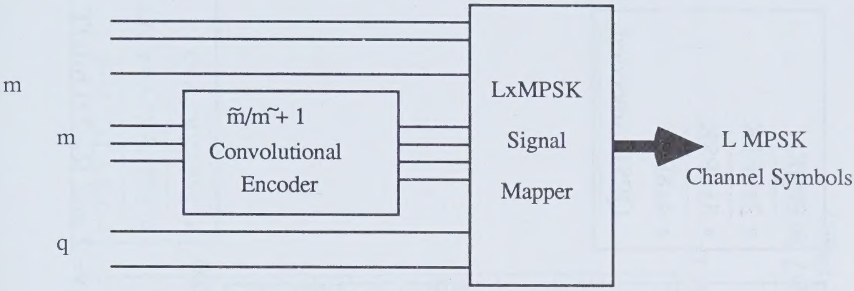

m
m̃/m̃+ 1
Convolutional
Encoder
m
LxMPSK
Signal
Mapper
L MPSK
Channel Symbols
q

Figure 1: Schematic Representation of a Multi-D Trellis Encoder

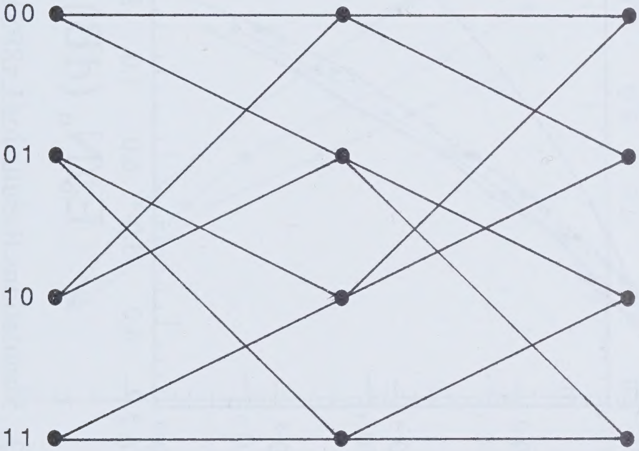


Figure 2: Trellis Diagram for the Rate 1/2 Convolutional Encoder with 4 States and Parity Polynomias 5 and 2.

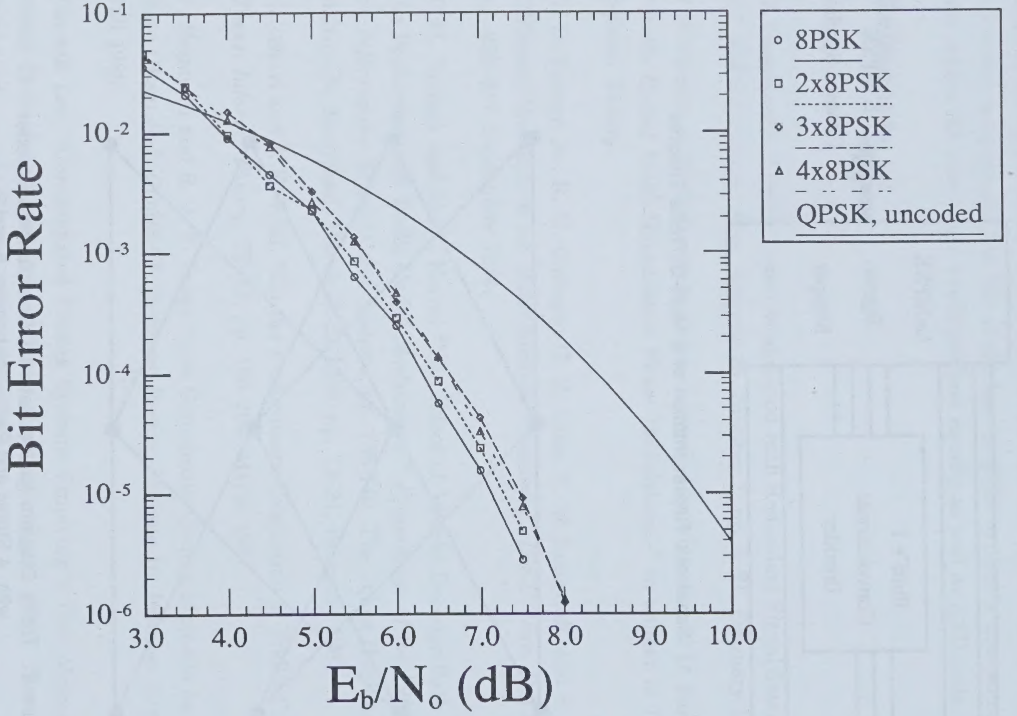


Figure 3: Simulations Results for Lx8PSK Codes with ν=2 and R=2.0 bits/T

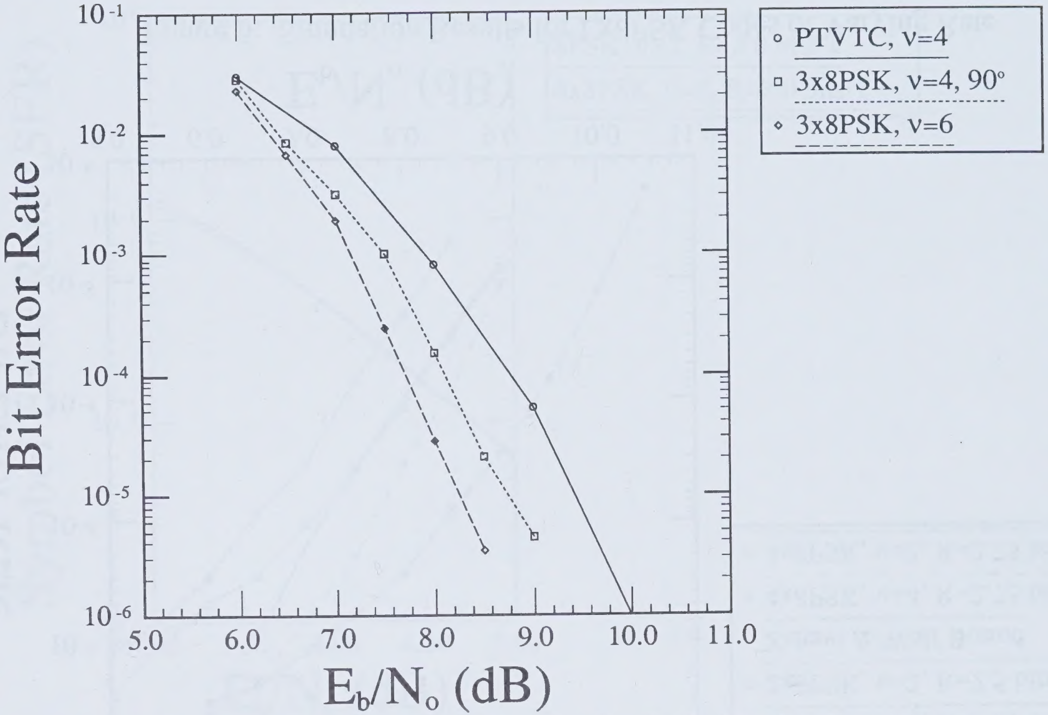


Figure 4: Simulation Results for R=2.67 bits/T.
(PTVTC results from Hemmati and Fang, 1986)

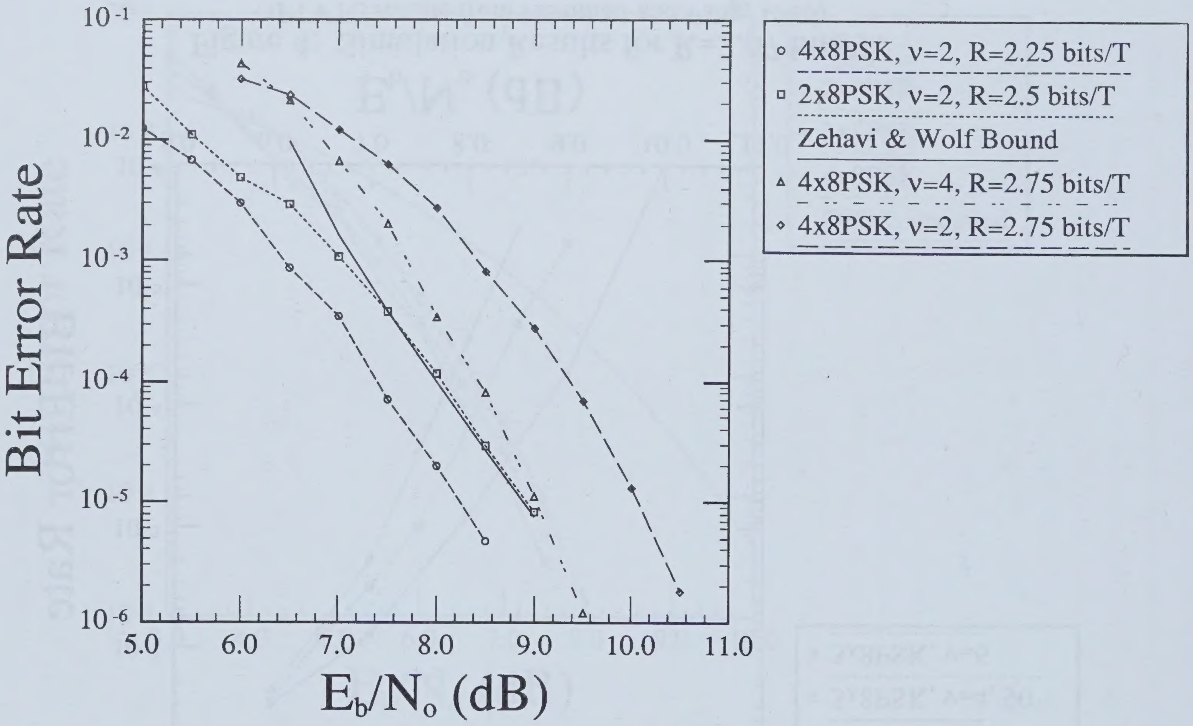


Figure 5: Simulation Results for Lx8PSK Codes of Varying Rate

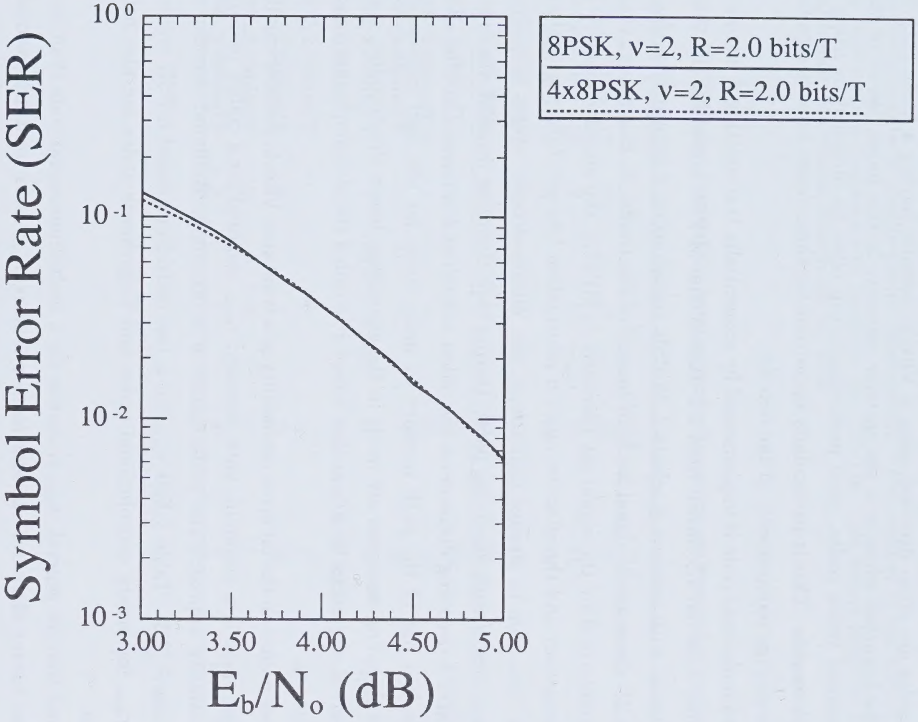


Figure 6: Symbol Error Rates for a RS(255,223) Outer Code

VITERBI DECODER PATH MEMORY AND MULTIDIMENSIONAL TRELLIS CODED MODULATION

JOSEPH M. NOWACK and MARK A. HERRO
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

Trellis codes are often decoded using a Viterbi decoder having a finite path length of at least 6ν branches where ν is the encoder memory. In this paper, we show that for multidimensional trellis codes, good performance can often be obtained by using much shorter path lengths. This is particularly important for these codes since it significantly reduces the storage requirements in the decoder.

A multidimensional code is implemented by sequentially transmitting L signals (each having a time duration T) chosen from a 2-dimensional M-PSK signal set [1]. A convolutional encoder with memory ν selects L M-PSK signals using k information bits. This produces a $2L$-dimensional signal for each branch of the trellis. In this paper, the codes will be characterized by the signal set (denoted LxMPSK), the encoder memory ν, the phase transparency and the effective rate (in information bits per 2-dimensional signal).

Assume that, due to storage limitations, the Viterbi decoder stores only the most recent τ branches of each surviving path. During each decoding interval, the truncated decoder makes a decoding decision on the oldest branch of k information bits, and then shifts these bits out of the path memory to make room for the next processed bits. Subsequent decoding decisions are made in the same way, hence the decoding decisions lag the received sequences by τ branches where τ is called the *decoder truncation length* or *path memory.*

An upper bound on the bit error probability of a truncated Viterbi decoder can be used to show that there is a minimum path memory, τ_{min}, required for a code's performance to asymptotically approach the performance of a maximum-likelihood decoder on the AWGN channel [2,3]. Table 1 lists τ_{min} for a few multidimensional 8-PSK codes. The values of τ_{min} for binary convolutional codes and Ungerboeck codes are also listed for comparison.

We found that, in general, τ_{min} is shorter for a multidimensional code than for other trellis codes having the same encoder memory. Also, τ_{min} for a multidimensional code is often much less than 6ν branches. Since k is typically large for these codes (e.g., 4-11), the number of bits stored per path, $k\tau$, can be reduced significantly by using the minimum path memory τ_{min} rather than 6ν branches.

[1] S. S. Pietrobon, R. H. Deng, A. Lafanechere, G. Ungerboeck and D. J. Costello, Jr., “Trellis Coded Multi-Dimensional Phase Modulation, ” to appear in *IEEE Trans. Inform. Theory.*

[2] F. Hemmati and D. J. Costello, Jr., “Truncation Error Probability in Viterbi Decoding, ” *IEEE Trans. Commun.*, **COM-25**, pp. 530–532, May 1977.

[3] J. M. Nowack and M. A. Herro, “The Effect of Viterbi Decoder Path Memory on the Performance of Trellis Coded Modulation ” *Proceedings of the 1989 Conference on Information Sciences and Systems*, pp. 735-740, The Johns Hopkins University, Baltimore, Maryland, March 22-24, 1989.

Table 1: Minimum truncation lengths for trellis codes with encoder memory ν.

Code	$\nu = 2$	$\nu = 3$	$\nu = 4$	$\nu = 5$	$\nu = 6$
Binary Conv. $\mathbf{R = 1/3}$	9	10	12	17	20
Binary Conv. $\mathbf{R = 1/2}$	8	11	17	21	28
Binary Conv. $\mathbf{R = 2/3}$	5	9	13	18	20
Binary Conv. $\mathbf{R = 3/4}$	-	9	-	16	24
Ungerboeck $\mathbf{8 - PSK}$	8	14	20	23	34
2x8PSK, 2.5bits/T	4	11	17	22	18(45°), 20(90°)
2x8PSK, 2.0bits/T	4	5	10	16	15
3x8PSK, 2.67bits/T	4	5	7	16	19
3x8PSK, 2.33bits/T	4	10(90°), 9(180°)	11	14	13(45°), 19(180°)
3x8PSK, 2.0bits/T	4	6	6	14(180°), 13(360°)	14
4x8PSK, 2.75bits/T	2	6	7	8	21
4x8PSK, 2.50bits/T	2	3	7	12	13
4x8PSK, 2.25bits/T	2	6	6	8	10
4x8PSK, 2.00bits/T	2	3	8	9	15(45°), 13(90°)

SOME NEW METHODS FOR SIMULATING SEQUENTIAL DECODERS USING IMPORTANCE SAMPLING

KHALED BEN LETAIEF
School of Electrical Engineering
Purdue University at West Lafayette
West Lafayette, IN 47907

JOHN S. SADOWSKY
School of Electrical Engineering
Purdue University at West Lafayette
West Lafayette, IN 47907

ABSTRACT

Importance Sampling is a classical technique for increasing the efficiency of Monte Carlo simulations. The basic principle is to simulate using a biased simulation distribution and then weight the observed simulation outputs to obtain unbiased estimates for the true distribution . When this technique is properly implemented, the result is an unbiased estimator which often requires orders of magnitude fewer simulation runs than conventional Monte Carlo. This paper presents two new importance sampling techniques for estimation of the distribution of computation for a sequential decoder employing the stack algorithm.

I. INTRODUCTION

Importance sampling (IS) is a Monte Carlo simulation technique which has been recognized as a powerful tool for simulating low probability events [1-5]. In many situations it can reduce by orders of magnitude the number of simulation runs required to obtain a specified accuracy. The basic idea behind importance sampling is to "bias" the *simulation distribution* in order to increase the relative frequency of the "important" events. This action is compensated by weighting the simulation data by a factor which is called the *importance sampling weight*. The end result is an unbiased estimator with a substantial variance reduction in comparison to conventional Monte Carlo.

In this paper we apply IS to the problem of estimating the distribution of computation of stack algorithm sequential decoders. We shall present two IS techniques which we shall refer to as the *reference path method* and the *partitioning method*. The reference path simulation method is an ad hoc technique based on distance properties of the code being simulated. The partitioning method, on the other hand, is motivated by an information theoretic ensemble averaging argument.

In section II we present some background on tree searching and the stack algorithm. In addition we shall develop a structural foundation for the stack algorithm simulation. Section III presents the importance sampling background for coded communications systems. This is done in the context of convolutional codes. Section IV presents our new importance sampling simulation schemes. Finally section V presents numerical results for the binary symmetric channel (BSC), as well as, the additive white Gaussian noise (AWGN) channel. Further discussions is found in the concluding section.

II. PRELIMINARIES

Tree searching and the Stack Algorithm

Consider a *tree*, that is, the usual directed graph consisting of *nodes* connected by *branches*. Starting from the unique *root node*, a *path* is a connected sequence of nodes. For each node there is a unique path which connects that node to the root node. The *depth* of a node shall refer to the number of branches on the path connecting that node to the root node. A path may be identified by its first node and the sequence of branches which connect the path nodes. We shall say node γ is a descendent of node δ if there exists a path which passes first through δ and then subsequently through γ. In this paper we will only consider binary trees corresponding to rate 1/2 convolutional codes. Generalization for rate k/n codes is straightforward.

Each branch in the tree will have a random weight called the *branch metric*. For each node γ, the *node metric* is the sum of branch metrics on the unique path connecting the root node to γ. The metric of the root node is zero. In particular, there is a unique *correct path*, and on this correct path the sequence of node metrics should tend on the average to increase; that is, the correct path node metric process has a "positive drift." Conversely, as we follow any path disjoint from the correct path the node metrics should tend to decrease; that is, an incorrect path has a "negative drift." Our goal is to identify the correct path. Given observation of the node metrics, the path most likely to be correct is the one which terminates with the maximum node metric value.

We are now ready to present the *stack algorithm* (also known as the Z-J algorithm). The *stack* is simply a list of previously examined nodes and their associated node metrics and it is ordered according to metric value. The *top-of-stack* node is the maximum metric node. The stack algorithm consists of the following steps:

1) Initialize the stack with the root node.
2) Compute the node metrics for each direct descendent of the top-of-stack node.
3) Remove the top-of-stack node from the stack and replace with its direct descendents.
4) Reorder the stack according to node metric values.
5) Test for termination. If not a terminal node, return to step 2 and continue.

Because the correct path metric process has a positive drift, and since all incorrect paths have a negative drift, the correct path nodes will tend to "float" to the top of the stack. However, it is entirely possible that the algorithm can from time to time mistakenly follow an incorrect path for some depth in the tree. This brings out one of the key characteristics of this algorithm, namely, the number of incorrectly hypothesized branching decisions per correct decision is a random variable.

The *correct node j*, shall refer to the (unique) node on the correct path at depth j. Now consider the *j'th incorrect subtree* $\mathcal{N}_j$; that is, the subtree of incorrect nodes on paths diverging from the correct path precisely at node j. $\mathcal{X}_j$ shall denote the subset of $\mathcal{N}_j$ which is actually hypothesized (that is, nodes which were on the stack at least once) by the algorithm. $\mathcal{X}_j$ is called the *j'th incorrect subset.* $\mathcal{X}_j$ is a random set, and

$$C_j = \text{the number of nodes in } \mathcal{X}_j$$

is a random variable. Let **F** be the *terminal path.* That is, the final hypothesized path in the tree. In this paper, we are interested in estimating the distribution of computation:

$$\mathcal{P}(C_j \geq M \mid E_j) \quad \text{for} \quad M \geq 1.$$

where E_j is the event { both node j and node j+1 are on **F** }.

A great deal of work has gone into the statistical analysis of the probability distribution of C_j [6-8]. The previous work however, was developed in the context of an "ensemble average" analysis of convolutional codes, and thus cannot predict the distribution of C_j for a specific code. Using this information theoretic approach, it has been demonstrated that the distribution of C_j has a *Pareto tail.* Specifically, $\mathcal{P}(C_j \geq M \mid E_j) \sim O(M^{-\rho})$ where ρ is the Pareto exponent [8, chapter 6].

In contrast to the ensemble average results of information theory just mentioned, here we consider the estimation of $\mathcal{P}(C_j \geq M \mid E_j)$ for specific codes.

A Format for the Simulation of the Stack Algorithm

Let $\mathbf{S}_j$ be the entire subtree emanating from j. We are only interested in estimating $\mathcal{P}(C_j \geq M \mid E_j)$ where C_j is the number of nodes examined in $\mathcal{N}_j$ which is a proper subtree of $\mathbf{S}_j$. It follows that our simulation schemes will be definitely much more efficient if only the data associated with the nodes in S_j are generated. Since the stack is ordered according to metric values only, the stack algorithm compares nodes of different tree depths in its decision process. Thus at a given instant, the stack may contain nodes in both S_j and S_j^c. Consequently, the search performed by the algorithm in S_j will be affected by the search performed in S_j^c before and after node j has been hypothesized, and therefore, to estimate $\mathcal{P}(C_j \geq M)$, knowledge of the entire history of the stack is required. However, if conditioned on the event E_j (i.e., that node j and node j+1 are on the terminal path), then this last statement is no longer true! In other words, *given the event* E_j, *then the search performed by the stack algorithm in* S_j *is not affected by the search in* S_j^c. This last statement is summarized in the following theorem. The proof is omitted for brevity.

The Fundamental Theorem: Let C_j and E_j be defined as above. Then for any $M \geq 1$

$$\mathcal{P}(\, C_j \geq M \mid E_j\,) \;=\; \mathcal{P}(\, C_0 \geq M \mid E_0\,).$$

As a result of the above theorem, we may estimate $\mathcal{P}(C_j \geq M | E_j)$ using multiple independent simulations, observing C_0 for each simulation, and then estimating $\mathcal{P}(C_0 \geq M | E_0)$ using the sample mean estimator.

Hereafter, we shall drop the conditioning in the notation: We simply write $\mathcal{P}(C_j \geq M)$ instead of $\mathcal{P}(C_j \geq M | E_j) = \mathcal{P}(C_0 \geq M | E_0)$.

We conclude this section by noting two important issues which, due to space limitation, will not be adequately discussed here. First, practical implementations of the stack algorithm always require a truncation strategy. Without such a strategy our simulations would still require an infinite amount of searching in S_j. For example, one common termination strategy is to delete any node from the stack if its metric is less than the top-of-stack node metric minus some threshold. The second issue is the "remerging phenomenon" which ultimately results in branching errors [8]. This phenomenon adds additional complexity to the simulation of the stack algorithm analysis as it corresponds to incorrect paths in the tree which behave exactly like the correct path after the point of remerging. These issues were properly accounted for in our simulation schemes but we shall not address them further here.

III. IMPORTANCE SAMPLING SIMULATION

Convolutional Codes

For brevity, we shall restrict our attention to rate 1/2 convolutional codes. For such codes, each time an information bit is shifted into the encoder, 2 bits are shifted out. Hence, the channel input symbols can be defined as blocks of 2 bits. Let K be the number of information bits stored in the encoder shift register not counting the most recent information bit input. The complexity of the code is determined by its *constraint length* which is equal to K+1. The *state* of the encoder is defined to be the last K information bits in the shift register. Since there are K bits which determine the state, the total number of states is 2^K. For each state at time k there are 2^K possible predecessor states, one corresponding to each of the information bit value (0 or 1) that could have been shifted out of the register during the state transition from time k–1 to time k. The encoder output at time k is thus a function of the encoder states at time k and k–1. Furthermore, note that because of its structure, it is clear that the encoder produces a binary tree code.

For a given sequence of information bits, let $\mathbf{u} = (u_1, u_2, \ldots)$ denotes the sequence of encoder output symbols. Given the sequence of channel output symbols $\mathbf{V} = (V_1, V_2, \ldots)$, the job of the decoder is to determine the information bit sequence which was most likely to have been transmitted, or equivalently, its corresponding path on the tree. For each branch on the tree, the decoder computes a *branch metric* $m(u_k,V_k)$ which is an indicator of the likelihood that u_k was indeed the channel input symbol which produced the random channel output symbol V_k. Throughout this paper, a "maximum likelihood like" metric known as the *Fano metric* [8] will be used to compute the branch metrics.

Importance Sampling

Consider a memoryless coding channel. At each time instant k, a channel symbol input u_k is transmitted over the channel to produce the channel output symbol V_k. Given u_k is transmitted, V_k is a random quantity with conditional $f_k(v_k|u_k)$. Since the channel is memoryless, the sequence of output symbols $\mathbf{V} = (V_1,V_2,\ldots,V_N)$ is a sequence of independent random channel output symbols with joint density

$$f(\mathbf{v}|\mathbf{u}) \;=\; \prod_{k=1}^{N} f_k(v_k|u_k) \tag{1}$$

If $f_k(v_k|u_k)$ does not depend on k, then the channel is also stationary.

Now for any $M \geq 1$, let $I_M(\cdot)$ be the indicator of the event $\{C_j \geq M\}$. Then the *importance sampling estimator* of $\mathcal{P}(\,C_j \geq M\,)$ is

$$\hat{P}_M \;=\; \frac{1}{L}\sum_{\ell=1}^{L} I_M(\mathbf{V}^{(\ell)})\, w(\mathbf{V}^{(\ell)}|\mathbf{u}) \tag{2}$$

with

$$w(\mathbf{v}|\mathbf{u}) \;=\; \frac{f(\mathbf{v}|\mathbf{u})}{f^*(\mathbf{v}|\mathbf{u})} \tag{3}$$

and $\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(L)}$ are independent random samples from the *importance sampling distribution* $f^*(\mathbf{v}|\mathbf{u})$. The likelihood ratio $w(\mathbf{v}|\mathbf{u})$ is called the *importance sampling weight.*

A straightforward computation indicates that the weight function specified by (3) does indeed produce an unbiased estimator; that is, $E^*[\hat{P}_M] = \mathcal{P}(\,C_j \geq M\,)$ when applied to (2). A similar computation indicates that the variance is

$$\begin{aligned} \mathrm{var}^*[\hat{P}_M] \;&=\; \frac{1}{L}\,\mathrm{var}^*[\,I_M(\mathbf{V})\,w(\mathbf{V}|\mathbf{u})\,] \\ &=\; \frac{1}{L}\left\{\, E^*\!\left[\,I_M(\mathbf{V})\left(\frac{f(\mathbf{V}|\mathbf{u})}{f^*(\mathbf{V}|\mathbf{u})}\right)^{2}\right] - \mathcal{P}(\,C_j \geq M\,)^2 \,\right\} \end{aligned} \tag{4}$$

where var^* [.] and E^*[.] denote the variance and expectation operations for the IS simulation distribution.

The expression (4) indicates that a good choice of the simulation density $f^*(\mathbf{v}|\mathbf{u})$ will tend to be large relative to $f(\mathbf{v}|\mathbf{u})$. This will tend to minimize (4), and hence, diminish the estimator's variance for a fixed L, or equivalently, reduce the number of simulation runs L for a given variance or accuracy. For further discussions about the choice of good IS distributions, the reader is referred to [2].

On a first exposure, it may seem reasonable to deduce that the best importance sampling models will be stationary and memoryless, especially if the true channel model is stationary and memoryless. However, our experience indicates that this is not the case. To properly understand this, first note that a good importance simulation model should tend to increase the relative frequency of the important error events. Thus, it should "bias" or force the stack algorithm to search excessively in the subtree $\mathcal{N}_j$. This suggests that we should begin our importance sampling simulations with a very noisy operating condition to ensure a high percentage of error events. However, if we continue the simulation runs with a noisy channel, then these simulations will tend to produce a lot of errors in other incorrect subtrees and the desired incorrect subset may become excessively large. Our desire is to emphasize only the "important events," that is, the kind of excessive searching in $\mathcal{N}_j$ that is most likely to actually occur. Thus, it is clear that the simulation channel model should start out with a high noise operating condition in order to initiate the error events of interest, and then in some fashion be programmed to become less noisy as time progresses.

V. THE IMPORTANCE SAMPLING SIMULATION SCHEMES

The Reference Path Method

Let $\mathbf{u}^c = (u_1^c, u_2^c, \ldots)$ be the encoder output sequence associated with the correct path. Likewise, let $\mathbf{u}_N^i = (u_1^i, u_2^i, \ldots, u_N^i)$ be the encoder output sequence associated with any incorrect path of depth N that diverges from the correct path at the root node. Next, let $d(\mathbf{u}^c, \mathbf{u}_N^i)$ be the *hamming distance* between the code sequences (up to depth N) associated with $\mathbf{u}^c$ and $\mathbf{u}_N^i$. The reason for our interest in this distance is that the "important" branching error events are precisely those with low to moderate distance $d(\mathbf{u}^c, \mathbf{u}_N^i)$. In other words, it is more likely for the stack algorithm to search incorrect paths with low distances than those with large distances.

Let $\mathbf{u}_N^r = (u_1^r, \ldots, u_N^r)$ be the encoder output sequence associated with a *reference path* of depth N. Such a path is chosen in such a way that its hamming distance from the correct path is minimized over all incorrect paths of depth N. The reference paths can be found by an exhaustive search for various depths of N.

Now for a given reference path, our idea is to design an importance sampling channel model which will tend to "trick" the stack algorithm into following the reference path instead of the correct path, at least up to depth N. Each simulation run will produce a randomly generated channel output sequence $\mathbf{V} = (V_1, V_2, \ldots)$ which will tend to cause reference path nodes to float to the top-of-stack position. However, we must bias the simulation distribution in such a way that the algorithm will ultimately make a correct decision at depth j+1. This is achieved by switching the importance sampling simulation channel model to the original channel model at depth N. As an example of such IS simulation models, consider the BSC with crossover probability ε. The simulation model is a non-stationary BSC with (time varying) crossover probability

$$\varepsilon_k^* = \begin{cases} 1/2 & \text{if } u_k^c \neq u_k^r \\ \varepsilon & \text{if } u_k^c = u_k^r \end{cases}$$

for $k \leq N$. For $k > N$ we use $\varepsilon_k^* = \varepsilon$. It should be easy to see that with roughly probability 1/2 the reference path will be examined up to depth k.

The Partitioning Method

In this section, we will discuss another scheme inspired by a general probability theory known as l*arge deviations theory* and an “ensemble average” Chernoff bound for variance (similar to the variance Chernoff bound used in [5].)

Suppose that the correct node j has been hypothesized and consider a fixed incorrect node δ which has depth j+n and is on an incorrect path diverging from the correct path at depth j. δ is a candidate node for $\mathcal{X}_j$. The kernel of our stack algorithm analysis is to consider the probability that δ is hypothesized; that is, $\mathcal{P}(\delta \in \mathcal{X}_j)$. The analysis is then simplified by comparing the correct path metric process with the metric process along a single fixed incorrect path emanating from node j. Now let,

$$S_n^c = \begin{array}{c}\text{metric of}\\ \text{correct node}\\ n+j\end{array} - \begin{array}{c}\text{metric of}\\ \text{correct node}\\ j\end{array} = \sum_{k=1}^{n} Z_k^c \tag{5}$$

where Z_k^c is the k+j'th branch metric on the correct path. Likewise, considering a fixed incorrect path in the j'th incorrect subtree, define

$$S_n^i = \begin{array}{c}\text{metric of}\\ \text{incorrect node}\\ n+j\end{array} - \begin{array}{c}\text{metric of}\\ \text{correct node}\\ j\end{array} = \sum_{k=1}^{n} Z_k^i \tag{6}$$

where Z_k^i is the k+j'th branch metric on the incorrect path. Define Γ to be the minimum metric value along the correct path metric after time j. We shall also define:

$$N_\Gamma^i \triangleq \min\{\, n : S_n^i \le \Gamma \,\} \tag{7}$$

The following lemma is a standard result which has been proven by several authors (see Lemma 6.2.1 in [8]).

Lemma: Consider a fixed incorrect path in the j'th incorrect subtree and let δ, S_n^c and S_n^i be defined as above. Then δ is never hypothesized if $S_m^i < \Gamma$ for some $m \le n$.

Now define N^* to be the depth at which the minimum value along the correct path (after depth j) occurs. Then we can partition the underlying probability space by the events $\{N^* = m\}$, m = 1,2, ... , and hence,

$$\mathcal{P}(C_j \ge M) = \sum_{m=1}^{\infty} \mathcal{P}(C_j > M \,;\, N^* = m). \tag{8}$$

For m = 1,2, ..., we bias the simulation model to estimate the terms in (8) separately. It turns out that only a few values of m are required to dominate the sum. The simulation model is

$$f_k^*(v_k|u_k) = \begin{cases} c_k \exp(\Psi(u_k,v_k))\, f_k(u_k|v_k) & \text{if } k \le m \\ f_k(u_k|v_k) & \text{otherwise} \end{cases}$$

where $\Psi(u,v)$ is an optimized twisting function of $\mathbf{u}_k$ and $\mathbf{v}_k$, and c_k is a normalization constant. Unfortunately, space limitation does not allow an adequate development of this twisting function. However, we can characterize this model by noting that it changes the correct path metric drift from a positive drift to a negative drift for $k \le m$, but then returns to the true positive drift for $k > m$. This clearly tends to produce a global minimum of the correct path metric process at depth m.

V. NUMERICAL RESULTS

This section demonstrates the potential of our simulations techniques. Figures 1 and 2 below present some simulation results for the rate 1/2 and constraint lengths 5 and 14 convolutional codes operating on the binary symmetric channel (BSC) the additive white

Gaussian channel (AWGN) respectively. These codes are sufficiently complex that their simulations using ordinary Monte Carlo is quite difficult, if not impossible, for low noise conditions.

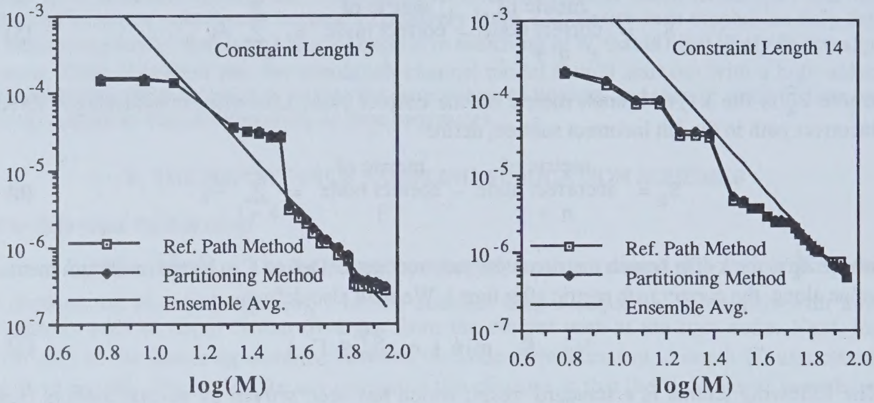


Figure 1: The distribution of computation estimates for the BSC with $\varepsilon = .005$.

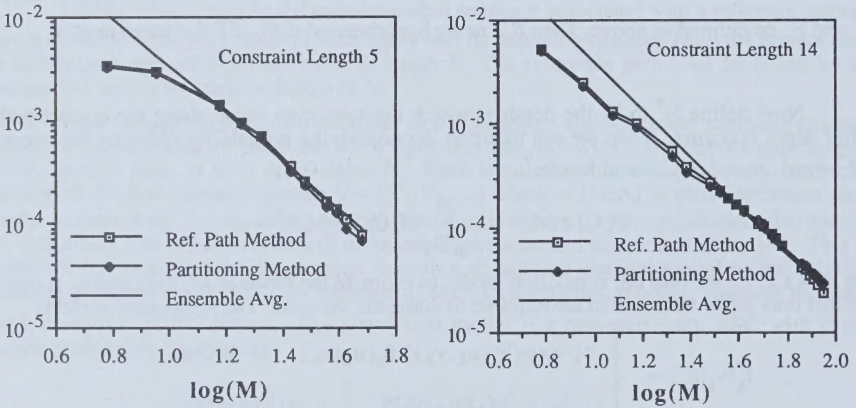


Figure 2: The distribution of computation estimates for the AWGN channel with the noise standard deviation = 0.6.

Recall that the information theoretic ensemble average analysis indicates that the distribution $\mathcal{P}(C_j \geq M)$ should have a Pareto tail with exponent ρ. Specifically, for the BSC of Figure 1 the ensemble average exponent is $\rho = 3.5$ and for the AWGN channel of Figure 2 the exponent is $\rho = 2.3$. For comparison, both Figures 1 and 2 also plot curves labeled "Ensemble Average." These curves are just $c\,M^{-\rho}$ where the constant c is fit for the tail of the our IS estimates. These curves appear as straight lines because these figures are logarithmic scale plots of the probability estimates against log(M).

The data in both Figures 1 and 2 was obtained empirically using a total of 300,000 simulation runs for both simulation schemes in the BSC case. In the AWGN case, a total of 600,000 simulation runs were used for the constraint 5 convolutional code. For the constraint 14 convolutional code, however, a total of 500,000 and 300,000 simulation runs were used for the reference path method and the partitioning method respectively. This is far more simulation runs than is required for estimating $\mathcal{P}(C_j \geq M)$ for M =1,..90; however, it was our desire to ensure accurate answers. In terms of CPU time (on a GOULD NP-1 computer), it was found that the reference path method usually takes less time than the partitioning method. This is reasonable because the partitioning method will tend to "grow" big incorrect subtrees as it tries to force the minimum of the correct path metric to occur at relatively large depths. For example, for the constraint 5 convolutional code operating on the BSC with ε=.005, the reference path scheme took 11 minutes, while the partitioning method took about 21 minutes. On the other hand, in terms of computational efficiency gain which is usually obtained by estimating the ratio of the numbers of simulation runs required for a specified precision, L_{MC} / L_{IS}, it was found that the partitioning scheme was overall better. For example, under the same conditions as in the previous example, the estimate for $\mathcal{P}(C_j \geq 90)$ was 0.3 x 10^{-6}. In this case, we estimated that $L_{MC} / L_{IS} = 10^6$ for the reference path method, and $L_{MC} / L_{IS} = 4.2 \times 10^7$ for the partitioning method. We should note that for a 10 % accuracy, ordinary Monte Carlo would require more than 3 x 10^8 simulation runs to estimate the above probability!

Table 1 below demonstrates the power and accuracy of our new simulation methods. The second and third columns of Table 1 give some of the estimates of the distribution of computation for the constraint 5 convolutional code operating on the BSC with ε =.005. Table 1 also lists estimates of the relative accuracy, that is, the estimator's standard deviation (as estimated by using the sample variance) as a percentage of the probability estimate.

Now recall that we are only interested in estimating $\mathcal{P}(C_j \geq M)$ for a given M. This implies that only the search performed by the stack algorithm in the j'th incorrect subtree $\mathcal{N}_j$ is of interest to us. Keeping this in mind, we conclude that any simulation scheme which modifies the stack algorithm so that only the incorrect nodes in $\mathcal{N}_j$ are extended will most likely be more efficient than any other simulation scheme which uses the stack algorithm. Because the search performed by the stack algorithm in $\mathcal{N}_j$ can be affected by the search performed by the stack algorithm in other incorrect subtrees, $\mathcal{N}_{j+1}$, $\mathcal{N}_{j+2}$, ..., it is possible that the estimates obtained using this "modified" stack algorithm might be incorrect. Table 2 and 3 below show that the difference is apparently insignificant. In other words, using a modified stack algorithm which operates exactly like the stack algorithm except that it 1) extends only the j'th incorrect subtree and 2) replaces every top-of-stack node which is on the correct path by only its direct descendent which is on the correct path, gives answers which are very close to the ones obtained when the stack algorithm is actually used. We will refer to such algorithm as the *modified stack algorithm simulation*.

Table 1: Comparison of the reference path (RM) and the partitioning (PM) methods for the constraint length 5 convolutional code operating on the BSC with $\varepsilon = .005$. $L = 300{,}000$ for both schemes.

	RM	Accuracy	PM	Accuracy
$\mathcal{P}(C_j \geq 12)$	$.14 \times 10^{-3}$	1.98 %	$.14 \times 10^{-3}$	0.93 %
$\mathcal{P}(C_j \geq 21)$	$.34 \times 10^{-4}$	2.30 %	$.36 \times 10^{-4}$	1.03 %
$\mathcal{P}(C_j \geq 33)$	$.27 \times 10^{-4}$	1.26 %	$.28 \times 10^{-4}$	1.06 %
$\mathcal{P}(C_j \geq 69)$	$.31 \times 10^{-6}$	3.26 %	$.38 \times 10^{-6}$	2.00 %
$\mathcal{P}(C_j \geq 90)$	$.28 \times 10^{-6}$	3.45 %	$.30 \times 10^{-6}$	1.98 %

Table 2: Comparison of modified stack algorithm simulation (MSA) and stack algorithm (SA) results for the constraint length 5 convolutional code operating on the BSC with $\varepsilon = .005$ using the reference path method. $L = 300{,}000$.

	MSA	Accuracy	SA	Accuracy
$\mathcal{P}(C_j \geq 3)$	$.540 \times 10^{-2}$	0.52 %	$.541 \times 10^{-2}$	0.57 %
$\mathcal{P}(C_j \geq 30)$	$.267 \times 10^{-4}$	1.25 %	$.272 \times 10^{-4}$	1.41 %
$\mathcal{P}(C_j \geq 45)$	$.148 \times 10^{-5}$	6.48 %	$.148 \times 10^{-5}$	6.51 %
$\mathcal{P}(C_j \geq 75)$	$.297 \times 10^{-6}$	3.34 %	$.295 \times 10^{-6}$	3.36 %

Table 3: Comparison of modified stack algorithm simulation (MSA) and stack algorithm (SA) results for the constraint length 14 convolutional code operating on the AWGN channel with noise standard deviation equal to 0.6. $L = 300{,}000$ and the reference path method was used.

	MSA	Accuracy	SA	Accuracy
$\mathcal{P}(C_j \geq 6)$	$.526 \times 10^{-2}$	0.68 %	$.523 \times 10^{-2}$	0.68 %
$\mathcal{P}(C_j \geq 27)$	$.346 \times 10^{-3}$	3.02 %	$.345 \times 10^{-3}$	3.07 %
$\mathcal{P}(C_j \geq 48)$	$.115 \times 10^{-3}$	4.67 %	$.112 \times 10^{-3}$	3.93 %
$\mathcal{P}(C_j \geq 78)$	$.369 \times 10^{-4}$	7.60 %	$.366 \times 10^{-4}$	6.95 %

By comparing the data in Table 2 and 3, it does appear that both the MSA and the stack algorithm give very close estimates and accuracy. In terms of CPU time we have found out that our simulation methods takes about twice less CPU time whenever the MSA is used instead of the stack algorithm.

VI. CONCLUSION

In this paper, we have presented two new importance sampling techniques for simulating (stack algorithm) sequential decoders. To illustrate the power and accuracy of these methods, we have presented some numerical results for the BSC and the AWGN channel. We have only considered the estimation of the distribution of computation. It is noted that bit error probability can be estimated using the method of [5].

The design of efficient and practical importance sampling simulation distributions is often ad hoc. Most of the important sampling analysis begins by proposing a family of candidate simulation distributions and then optimizing some parameters of the family. In this paper, the reference path method was an ad hoc simulation technique obtained by application of specific knowledge about the (stack algorithm) sequential decoder operation. The partitioning method on the other hand was developed using large deviations theory. Because large deviations theory is not limited to ideal channels, the partitioning method promises to be a powerful tool for performance evaluation of sequential decoders for non-ideal channels. In a future work we shall present our large deviations analysis of simulation of the stack algorithm.

REFERENCES

[1] K. S. Shanmugan and P. Balaban, "A modified Monte Carlo simulation technique for the evaluation of error rate in digital communication systems," *IEEE Trans. Commun.*, vol. COM-28, pp. 1916-1924, 1980.

[2] P. Bratley, B. L. Fox and L. E. Schrag, *A Guide to Simulation.* New York: Springer, 1987.

[3] D. Lu and K. Yao, "Improved importance sampling technique for efficient simulation of digital communications systems," *IEEE J. Select. Areas Commun.*, vol. SAC-6, pp. 67-75, 1988.

[4] J. S. Sadowsky, "A new method for Viterbi decoder simulation using importance sampling," to appear in *IEEE Trans. Commun.*

[5] J. S. Sadowsky, J. A. Bucklew, and P. Ney, "Monte carlo simulation and large deviations theory for uniformly recurrent markov chains," to appear in *Journal of Applied Probability*, March 1990.

[6] I. M. Jacobs and E. R. Berlekamp,"A lower bound to the distribution of computation for sequential decoding," *IEEE Trans. Inf. Theory*, IT-13, pp. 167-174, 1967.

[7] F. Jelinek, "An upper bound on moments of sequential decoding effort,"*IEEE Trans. Inf. Theory*, IT-15, pp. 140-149, 1969.

[8] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding.* New York: McGraw Hill, 1979.

A SYMBOL ERROR UPPER BOUND FOR CONVOLUTIONAL CODES

Robert J. McEliece and Ivan M. Onyszchuk
California Institute of Technology, Pasadena, CA 91125

The probability that a b-bit symbol is decoded incorrectly, SER_b is the performance measure of a convolutional code when it is concatenated with an outer block code having symbols from $\mathrm{GF}(2^b)$. On a binary-input, output-symmetric DMC [1], a maximum likelihood decoder's bit error rate (BER) is bounded by $\mathrm{SER}_1 \leq \sum_{d=d_\mathrm{f}}^{\infty} i(d)P_\mathrm{d}$ (1) where $i(d)$ is the total weight of all information sequences producing weight d "fundamental" trellis paths: those from decoder state 0 into state 0 via nonzero states. P_d is the probability that the decoder outputs a fundamental path having distance d from the one transmitted and d_f is the code's free distance.

Let $\underline{u}$ be a length j binary vector which begins and ends with a 1 and contains i ones but not m consecutive zeroes. When $\underline{u}$ followed by m zeroes is input to a memory m, rate $1/n$, binary convolutional encoder containing zeroes, the output $\underline{c}$ is a weight d, length $\ell = j+m$ trellis branches fundamental path. Suppose that (wlog) $\underline{0}$ is transmitted over a DMC. For each decoded bit, there are i mutually exclusive ways, each occurring with probability P_d, that $\underline{c}$ could be decoded instead of $\underline{0}$, causing a 1 output bit instead of 0. Therefore, the bit error rate is bounded by (1).

Note. *All $j+b-1$ possible ways that a length j information vector $\underline{u}$ could overlap a b-bit decoded symbol will cause an error when $\underline{c}$ is chosen instead of $\underline{0}$* **unless** *$\underline{u}$ contains at least b consecutive zeroes or equivalently $b \leq m-1$ and $j \geq b+2$.*

Since $\underline{u}$'s of length $j = \ell - m \geq b+2$ usually produce weight $> d_\mathrm{f}$ fundamental paths, the decoder will chose these paths over $\underline{0}$ with very low probability as compared to fundamental paths having weight near d_f and whose $\underline{u}$'s are short (i.e. 1,11,101,111). Also, each sequence of b consecutive zeroes in $\underline{u}$ will decrease by only 1 the number of ways that $\underline{u}$ might cause a symbol error. For example, if $m = 6$ and $b = 4$, then $\underline{u} = 100001$ (whose $\underline{c}$ has weight $16 > d_\mathrm{f} = 10$ in the rate 1/2 NASA code) can cause a 4-bit symbol error in 8 instead of 9 ways. Therefore, the following upper bound (2) based upon the above fact is as tight as the bit error rate bound (1), as verified for $b = 4$ and $b = 8$ by simulations of the $m = 6$, rate $R = 1/2$ NASA code on the unquantized AWGN channel (AGC). Since $\underline{u}$ could corrupt each b-bit symbol in $\leq \ell - m + b - 1$ ways,

$$\mathrm{SER}_b \leq \sum_{d=d_\mathrm{f}}^{\infty} \sum_{\ell=m+1}^{\infty} (\ell - m + b - 1)a(d,\ell)P_\mathrm{d} = \sum_{d=d_\mathrm{f}}^{\infty} \left[[b-1-m]a(d) + l(d)\right] P_\mathrm{d} \qquad (2)$$

where $a(d,\ell)$ is the number of weight d, length ℓ fundamental trellis paths, $a(d)$ is the number of weight d fundamental paths and $l(d)$ is their total length in trellis branches. In terms of the code's complete path weight enumerator $T(D, L, I)$ [1],

$$\mathrm{SER}_b \leq \frac{\partial}{\partial L}\left[L^{b-1-m}T(D,L,I)\right]_{L=I=1,D=\gamma} = (b-1-m)T(D) + \left.\frac{\partial T(D,L)}{\partial L}\right|_{L=1,D=\gamma}, \qquad (3)$$

where $\gamma = \sum_{y \in Y} \sqrt{p(y|0)p(y|1)}$ and Y is the channel output alphabet. For the AGC, $P_d = Q(\sqrt{2dRE_b/N_0}) \leq Q(\sqrt{2d_\mathrm{f}RE_b/N_0})e^{(d_\mathrm{f}-d)RE_b/N_0} = ke^{-dRE_b/N_0} = k\gamma^d$, where

---

This work was partially supported by Caltech's Jet Propulsion Laboratory, and by the Air Force Office of Scientific Research, under contract AFOSR-88-0247.

$Q(x) = \int_x^\infty e^{-y^2/2} \frac{dy}{\sqrt{2\pi}}$ [1, p. 247]. The SER_b bounds (3) diverge at $E_b/N_0 = -R^{-1} \ln z$ (1.74 dB for the NASA code), where z is the pole of smallest magnitude in $T(D)$. On the AGC at very high bit signal-to-noise ratio E_b/N_0

$$\mathrm{SER}_b \;\sim\; [(b-1-m)a(d_\mathrm{f}) + l(d_\mathrm{f})]Q(\sqrt{2d_\mathrm{f} RE_b/N_0}).$$

$s_{d_\mathrm{f}}(b) = (b-1-m)a(d_\mathrm{f}) + l(d_\mathrm{f})$ (> 0 for *any* $b \geq 1$ and $m \geq 1$ because $\ell \geq m+1$) is the number of ways that weight d_f fundamental paths can corrupt a b-bit symbol.

For punctured rate $(n-1)/n$ codes derived from rate 1/2 convolutional codes, one can verify that the SER_b bound (2) holds if the right side is multiplied by $1/(n-1)$ and m and ℓ refer to the rate 1/2 code memory and punctured trellis branches.

Now suppose that a convolutional code is concatenated with an outer block code having b-bit symbols, rate $R_\mathrm{bl} = k/n$ and minimum distance d_min. If $e = \lfloor (d_\mathrm{min}+1)/2 \rfloor$, the concatenated decoder bit error probability is

$$P_\mathrm{bit} = \frac{\mathrm{SER}_b}{\mathrm{BER}} \sum_{j=e}^{n} \binom{n}{e} (\mathrm{SER}_b)^j (1 - \mathrm{SER}_b)^{n-j} \, \frac{j}{n} \, .$$

$$P_\mathrm{bit} \;\sim\; c \binom{n-1}{e-1} (\mathrm{SER}_b)^e \;\approx\; c \; s_{d_\mathrm{f}}(b)^e \binom{n-1}{e-1} \left[Q(\sqrt{2d_\mathrm{f} RR_\mathrm{bl} E_b^*/N_0}) \right]^e$$

where $E_b^*/N_0 = R_\mathrm{bl}^{-1} E_b/N_0$ is the concatenated decoder operating bit signal-to-noise ratio and $c = [(b-1-m)a(d_\mathrm{f}) + l(d_\mathrm{f})]/i(d_\mathrm{f})$. Since $Q(x) < \frac{e^{-x^2/2}}{x\sqrt{2\pi}}$ for $x > 0$, the concatenated code free distance is $e \cdot d_\mathrm{f} = d_\mathrm{f} \lfloor (d_\mathrm{min}+1)/2 \rfloor$ as expected.

We computed the complete path weight enumerator $T(D, L, I)$ for the $m = 6$, rate 1/2 NASA code (among many others) which for $L = I = 1$ is

$$\frac{\begin{array}{l} 11D^{10} - 6D^{12} - 25D^{14} + D^{16} + 93D^{18} - 15D^{20} - 176D^{22} - 76D^{24} + 243D^{26} + 417D^{28} \\ -228D^{30} - 1156D^{32} - 49D^{34} + 2795D^{36} + 611D^{38} - 5841D^{40} - 1094D^{42} + 9575D^{44} \\ +1097D^{46} - 11900D^{48} - 678D^{50} + 11218D^{52} + 235D^{54} - 8068D^{56} - 18D^{58} + 4429D^{60} \\ -20D^{62} - 1838D^{64} + 8D^{66} + 562D^{68} - D^{70} - 120D^{72} + 16D^{76} - D^{80} \end{array}}{\begin{array}{l} 1 - 4D^2 - 6D^4 - 30D^6 + 40D^8 + 85D^{10} - 81D^{12} - 345D^{14} + 262D^{16} + 844D^{18} - 403D^{20} \\ -1601D^{22} + 267D^{24} + 2509D^{26} + 389D^{28} - 3064D^{30} - 2751D^{32} + 2807D^{34} + 8344D^{36} \\ -1960D^{38} - 16133D^{40} + 1184D^{42} + 21746D^{44} - 782D^{46} - 21403D^{48} + 561D^{50} + 15763D^{52} \\ -331D^{54} - 8766D^{56} + 131D^{58} + 3662D^{60} - 30D^{62} - 1123D^{64} + 3D^{66} + 240D^{68} - 32D^{72} \\ +2D^{76} \end{array}}$$

To compute the SER_b bounds, the terms (corresponding to the 6 least magnitude poles of $T(D)$ above) in the partial fraction expansions of $\partial T(D, L, I)/\partial L$ and $\partial T(D, L, I)/\partial I$ at $L = I = 1$ were used to obtain the approximations (with a relative error < 0.0001 for $k > 4$) :

$$\begin{aligned} a(2k+10) &\approx 6.82(2.3876225)^{2k} + 4.25(1.657193)^{2k} \cos(0.310 - 1.967k) \\ l(2k+10) &\approx (77.725 + 22.625k)(2.3876225)^{2k} + 39.3(1.657193)^{2k} \cos(0.485 - 1.967k) \\ &\quad + (2k+1)7.2676(1.657193)^{2k} \cos(0.383 - 1.967k) \\ i(2k+10) &\approx (24.474 + 12.018k)(2.3876225)^{2k} + 9.942(1.657193)^{2k} \cos(0.575 - 1.967k) \\ &\quad + (2k+1)2.8723(1.657193)^{2k} \cos(0.366 - 1.967k) \end{aligned}$$

[1] A. J. Viterbi and J.K. Omura, "Principles of Digital Communication and Coding," McGraw-Hill, 1977.

A Programmable Systolic BCH Decoder

John J. Vaccaro, Thomas M. Hopkinson, & Kevin C. Moore
MITRE Corporation
Bedford, MA

Abstract

This paper presents a programmable BCH error-correction decoder that supports processing over $GF(2^m)$, $m \leq 8$ where the user can specify the blocklength n, the error correction capability t, and the field generator polynomial. The decoder provides both error and erasure decoding for all codes with rates greater than 1/2. The decoder consists of a systolic processor and a control processor, with a single chip systolic processor decoding any code with an error-only correction capability up to $t = 16$. These systolic processors can be arrayed to provide any desired error correction capability. The focus of the paper is on a new systolic technique for solving the key equation that allowed the development of this completely extendable array.

Introduction

Although BCH codes are powerful and efficient, their decoding algorithms are computationally intense finite-field procedures that require special-purpose hardware for high-speed area-efficient implementations. Flexible high-performance decoders must be developed, implemented, and made available so that system engineers can concern themselves only with code performance and not implementation issues. Such decoders will minimize the cost and risk of employing powerful error-correction techniques.

A programmable BCH error-correction decoder that can support processing over $GF(2^m)$, $m \leq 8$ has been implemented in VLSI technology. The decoder provides both **error and erasure** decoding for all codes with rates greater than 1/2. The decoder is implemented in 2 micron CMOS technology and supports an information bit data rate exceeding 20 mega-bits per second. The decoder consists of a systolic processor and a control processor. A single chip systolic processor can support any code with an error-only correction capability up to $t = 16$. With erasures a fixed hardware system will have a reduced maximum error correction capability. This systolic processor chip is composed of five individual systolic arrays. However, this chip is itself systolic, meaning that the five individual systolic arrays can be extended by arraying chips together, providing any desired error correction capabilty.

The major contribution of this work is a new implementation technique for solving the key equation; the focus of this paper is the mapping of Eastman's algorithm and architectural ideas into a systolic processor [4]. Based on this key equation processor we have implemented a programmable decoder in which the user can specify the blocklength n, the error correction capability t, and the field generator polynomial.

Approach

An evaluation of several decoding architectures resulted in the decoder architecture presented in figure 1. There are five major components: 1) a syndrome calculator, 2) the key equation solver, which yields an error-locator polynomial, 3) a Chien search to identify zeros of the error-locator polynomial, 4) a recursive extender to produce a transform-domain error sequence and 5) a limited point inverse transform. A complexity analysis indicated that, for rates greater than or equal to 1/2, the proposed system, which is based on our key equation algorithm [4], is more efficient than complete inverse transform techniques and techniques utilizing Forney's algorithm [6]. A folded 1-D architecture is presented that supports code rates greater than 1/2. Arbitrary rate codes are supported by not folding the 1-D array.

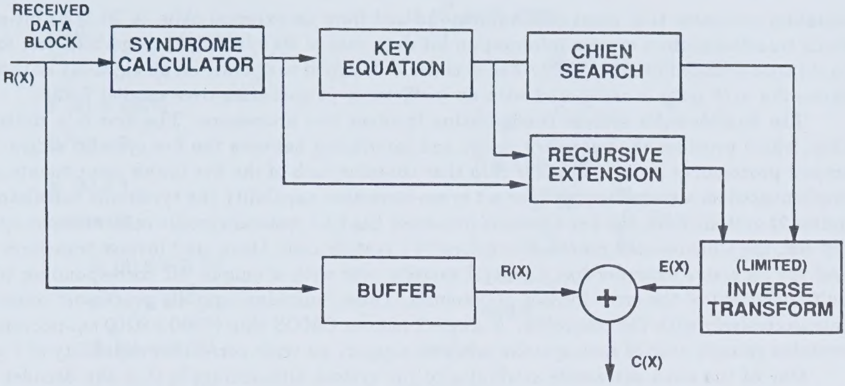


Figure 1. Decoder Architecture

These five components can all be realized as systolic arrays. The syndrome calculator, inverse transform, and Chien-search processor are all partial number theoretic transforms; each can be evaluated via Horner's scheme, which results in a systolic cell that is based on a multiply-and-accumulate operation. A systolic technique for evaluating the recursive relation (this is the recursive-extension function)

$$E_k = \sum_{l=1}^{\nu} \Lambda_l E_{k-l}$$

has been presented by Demassieux et al. [3]. The value ν is the number of actual errors present in a received block.

The solution of the key equation is the most difficult of the decoding operations and has received considerable research activity. Recent focus has been on developing algorithms that are suited for VLSI implementation [3],[7],[8]. A group at the Jet Propulsion Laboratory has developed a decoder [8] where the key equation is solved using a greatest common divisor (GCD) implementation developed by Brent and Kung [2]. This systolic GCD algorithm requires $2t$ cells and requires only local communication. In addition, there is no need for inverse field element calculations. However, every cell must compute several logical control signals and the resulting systolic cell is not regular. The technique that we have developed requires only $t+1$ cells, and easily handles decoding with erasures. Liu has proposed solving the key equation by a direct implementation of the Berlekamp-Massey algorithm [7]. This approach requires an inverse field element calculation within every iteration and a global broadcast of the discrepancy. These are both negative features, and the global broadcast is potentially the limiting factor in the performance of the system.

Our architecture for solving the key equation is based on the recent work of Eastman [4] that developed a new systolic approach that eliminates the need for field-element division and removes the need to calculate a vector inner-product (discrepancy) during the iterative portion of the decoding process.

Each of the five systolic processors is implemented as a bit-serial processor. Each systolic cell is built from simple bit-slices, providing a highly regular easy to implement architecture. The arithmetic computations require no ripple between bit-slices, so that the speed of the system is equivalent to the speed of an individual bit-slice, which is limited only by the speed of a four-input exclusive-or gate. Actually the worst case delay involves a bit-slice of the key equation processor that must communicate to and from an external chip. A 20 MHz bit-serial clock translates into a system information bit data rate of $20(k/n)(m/8)$ mega-bits/sec. for an (n,k) code decoded over $GF(2^m)$. The decoder is designed to operate on an eight-bit arithmetic cycle; the $m/8$ term is associated with an inefficiency in operating over smaller fields.

The final decoder system configuration involves two processors. The first is a controller chip, which provides the necessary clocks and interfacing between the five systolic arrays. The second processor is a single decoder chip that contains each of the five major components, each implemented as a systolic array. For a t error-correction capability the syndrome calculator requires $2t$ systolic cells, the key equation processor has $t+1$ general systolic cells with one special left cell, the Chien search processor requires $t+1$ systolic cells, there are t inverse transform cells and the recursive extender has t general systolic cells with a unique left corresponding to the unit coefficient of the error locator polynomial. These individual systolic processors communicate exclusively with the controller. A single 2 micron CMOS chip (7900×9200 sq.-micron die) contains enough cells of each systolic array to support an error-correction capability of $t = 16$.

One of the most attractive attributes of the system architecture is that the decoder chip itself is systolic. Each of the individual systolic arrays is extended by arraying chips together, so that any desired error-correction capabilty can be attained by using multiple decoder chips with a single controller.

The Key Equation Processor

This section describes an algorithm and its circuit implementation for solving the key equation. We begin by presenting a technique for error only decoding and then extend this to include erasures.

Introduction

Given $2t$ syndromes, S_j: $j = 0, 1, \ldots 2t-1$, a syndrome polynomial is defined as

$$S(x) = \sum_{j=0}^{2t-1} S_j x^j .$$

The first root of the generator polynomial has been chosen to be α^0; the system naturally supports the choice of any first root. The error locator polynomial is defined as

$$\Lambda(x) = \prod_{l=1}^{\nu}(1 - X_l x) = 1 + \sum_{i=1}^{\nu} \Lambda_i x^i ,$$

where the X_l are the error locations and the zeros of this polynomial are related to the inverse error locations [1]. The polynomials $\Lambda(x)$ and $S(x)$ satisfy the recursion

$$-S_{j+\nu} = \sum_{i=1}^{\nu} S_{j+\nu-i}\Lambda_i ,$$

where $j = 0, 1, 2, \ldots, \nu - 1$. The solution of the key equation

$$\Omega(x) = \Lambda(x)S(x) \pmod{x^{2t}}$$

provides the Λ_i's that satisfy the recursion.

Given the syndromes S_i, the program in figure 2 will calculate $b^N(x)$ which is a scaled version of the desired polynomial $\Lambda(x)$, ($b^N(x) = \gamma\Lambda(x)$) [4],[5]. The discrepancy for the jth iteration is simply r_j^N, where r_j^N is the jth coefficient of the $r^N(x)$ polynomial. Eastman's algorithm successively zeros out the lowest coefficient of $r^N(x)$ during each iteration, so that after the jth iteration the least significant j coefficients are zero.

There are four polynomials in Eastman's error-only algorithm, $b^O(x)$, $b^N(x)$, $r^O(x)$, and $r^N(x)$, ($b^T(x)$ and $r^T(x)$ are temporary polynomials). The polynomial $b^N(x)$ is akin to

```
$l = 0$
$j = 0$
$r^O(x) = 1$
$r^T(x) = xS(x)$
$b^O(x) = 0$
$b^T(x) = 1$
while (1)
    $b^O(x) = x\, b^O(x)$
    $r^O(x) = x\, r^O(x)$
    $b^N(x) = b^T(x)$
    $r^N(x) = r^T(x)$
    j++
    if ($j > 2t$)   break ;
    if ($r_j^N$ != 0) then
        $b^T(x) = r_j^O\, b^N(x) - r_j^N\, b^O(x)$
        $r^T(x) = r_j^O\, r^N(x) - r_j^N\, r^O(x)$
        if ($j > 2l$) then
            $b^O(x) = b^N(x)$
            $r^O(x) = r^N(x)$
            $l = j - l$
        end if
    end if
end while
```

Figure 2. Eastman Key Equation Algorithm

Berlekamp's generated shift register connection polynomial, and $b^O(x)$ is the old shift register that created a discrepancy. A new shift register $b^T(x)$ must be created when the current shift register $b^N(x)$ fails to predict the next element of the recursion. The previous shift register $b^O(x)$ had a discrepancy $r_j^O(x)$ and the new shift register is constructed from the two previous shift registers as

$$b^T(x) = r_j^O b^N(x) - r_j^N b^O(x).$$

The Eastman 2-D Array

Eastman mapped the algorithm of figure 2 into a two-dimensional systolic array consisting of $2t$ rows, where data moving between all cells is clocked. Figure 3 shows the history of a single solution to the key equation, with the results produced by each cell shown within the cell. With the author's permission figure 3 was taken from [4], and is a specific example over GF(11), a favorite field of Eastman's. Each row in the 2-D array consists of a single special left cell (figure 4) and many identical general cells (figure 5). The delay in figure 4 is required because the left-most general cell must wait for its upper right neighbor to complete before it can execute.

Looking at the general cell, the $r^N(x)$ polynomial is shifted left between each level, equivalent to the position j incrementing in the program of figure 2. The shift occurs because the output of row i, $\tilde{r}_k^N$, becomes the input r_k^N at row $i+1$. The $(k-1)$st coefficient of the polynomial $r^N(x)$ is r_{k-1}^N. In the algorithm the polynomial $r^O(x)$ is shifted right during each iteration to force the previous discrepancy to be in the jth position. The algorithm shifts $r^O(x)$ to the right between each loop (or stage), but the cell inherently shifts it left, so that the net effect is that the polynomial stays in place. The shift right is effected by having the output $\tilde{r}_k^O$ as a function of the coefficients of one degree less, r_{k-1}^O and r_{k-1}^N.

The special left cell (figure 4) provides the current and previous discrepancies to the entire row along with the control signal e which tells the cell that a shift register length change has occurred, requiring the old polynomials to be updated, $r^O(x) \leftarrow r^N(x)$ and $b^O(x) \leftarrow b^N(x)$.

With a 2-D array the hardware can simultaneously process $5t$ codewords. A particular problem is in a single column every other cycle. This fact will be exploited later to compress this two-dimensional array into one dimension.

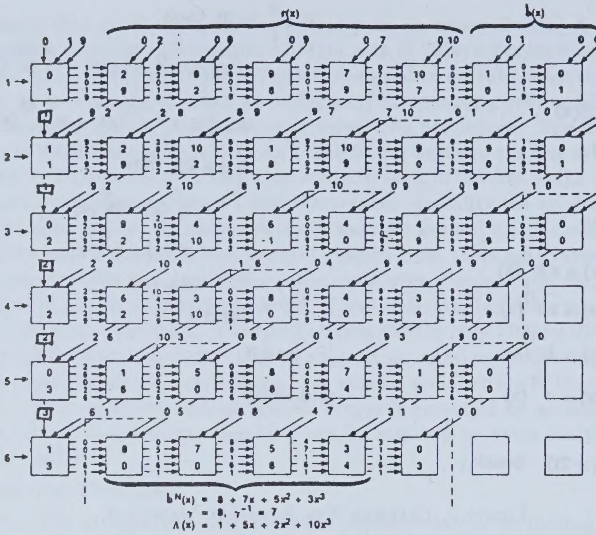

r(x)
b(x)
1
2
3
4
5
6
b^N(x) = 8 + 7x + 5x^2 + 3x^3
γ = 8, γ^-1 = 7
Λ(x) = 1 + 5x + 2x^2 + 10x^3

Figure 3. Eastman 2-D Array

$$(r^O_k)_{out} = (r^O_j)_{in}$$

$$(r^N_k)_{out} = (r^N_j)_{in}$$

$$(r^O_j) = (r^O_j)_{in}$$

$$(r^N_j) = (r^N_j)_{in}$$

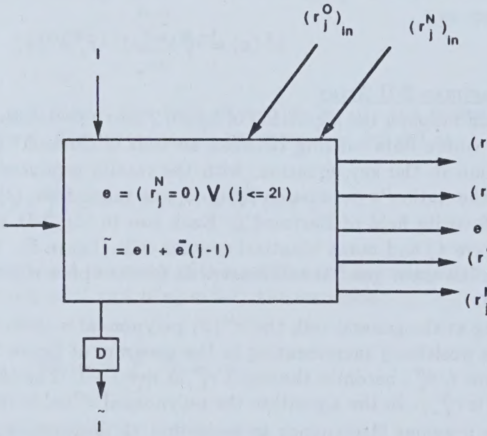

$(r^O_j)_{in}$
$(r^N_j)_{in}$
l
j
$e = (r^N_j = 0) \vee (j <= 2l)$
$\tilde{l} = el + \bar{e}(j-l)$
$(r^O_k)_{out}$
$(r^N_k)_{out}$
e
(r^O_j)
(r^N_j)
D
$\tilde{l}$

Figure 4. Special Left Cell

If $5t$ syndrome polynomials are available then it is possible to keep all cells in the array 100 percent utilized. This 2-D capability provides a tremendous decoding throughput. In a single channel system the data will arrive serially and the syndromes can be computed sequentially; it only becomes necessary for the key equation processor to complete before a new syndrome

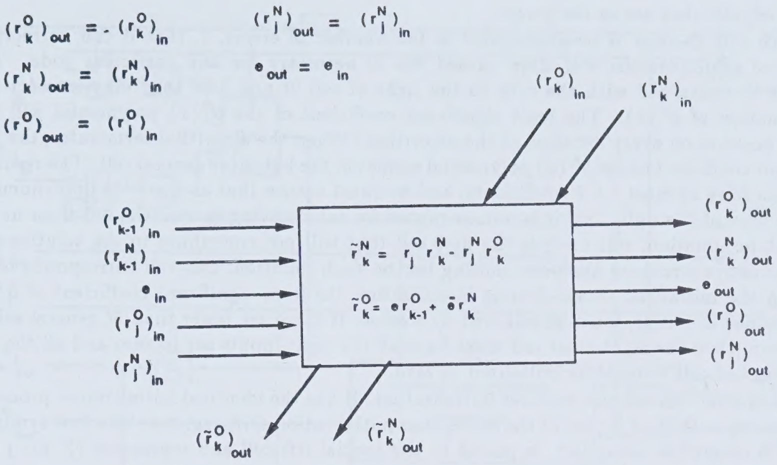


Figure 5. General Cell at Stage j

polynomial is computed. In this case there will be many idle cells and the 2-D array utilization is extremely low.

We are interested in $b^N(x)$ as it exits the bottom row of the array. As the data is shifted left on each successive stage, we must prevent the two polynomials $b^N(x)$ and $r^N(x)$ from interfering with each other. There are $2t$ iterations, or stages in the algorithm, and it can be shown that there are at most t length changes for the $r^N(x)$ polynomial. Thus, the polynomial $r^N(x)$ can grow by at most t coefficients. The polynomial $b^N(x)$ will shift left one position every stage of the architecture. Since the $r^N(x)$ polynomial is used to provide discrepancy values for each of the $2t$ iterations of the algorithm, the coefficients of $r^N(x)$ greater than $2t$ are never used. Thus it is only necessary to maintain the position of the $(2t)$th coefficient of $r^N(x)$ so that $b^N(x)$ can begin in the adjacent cell. It is only necesary that the r^O_{k-1} and r^N_{k-1} inputs to the least significant cell of the $b^N(x)$ polynomial both be set to zero. This is easily done algorithmicaly, and is shown in figure 3, where a dashed line separates the r and b polynomials.

A 1-D Array (error only)

Serial communication systems will naturally generate syndromes in a serial manner. Within each column of the 2-D array each problem appears at most once, suggesting that the 2-D array can be compressed into a single row, which will require $2t$ iterations to solve a single key equation problem. Assume that at time τ cell i is operating on iteration v, (i.e. row v in the 2-D case). Then at time $\tau + 1$ cell $i + 1$ operates on its iteration v, since it required inputs from cell i. However, cell i cannot execute its iteration $v + 1$ until cell $i + 1$ completes its iteration v. Thus cell i executes iteration $v + 1$ at time $\tau + 2$, and each cell is seen to be utilized 50 percent of the time. In a straight-forward implementation this corresponds to a total compute time of $4t$ for each cell, since each cell is only active 50 percent of the time on a given problem.

The syndromes are input in a serial fashion on a dedicated input shift-register to initialize the special left cell and the left-most $2t - 1$ general cells. The most significant coefficient of $S(x)$ leads the way finally ending up in the $(2t - 1)$st general cell. The special left cell gets initialized with the least significant coefficient of $S(x)$. The polynomial $b^N(x)$ initialization begins in cell $2t$, this cell receives the least significant coefficients of $b^N(x) = 1$ and $b^O(x) = 0$ (see the initialization loop in figure 2). Initialization to the right of cell $2t$ depends upon the number of cells that are in the array.

Since this decoder is programmable in the number of errors, t, that it can correct, the number of available cells will often exceed the $2t$ necessary for any particular code. Thus we must be concerned with the cells to the right of cell $2t$ and how they may interact with the formation of $b^N(x)$. The least significant coefficient of the $b^N(x)$ polynomial will shift left one position on every iteration of the algorithm. When the algorithm terminates, the least significant coefficient of the $b^N(x)$ polynomial resides in the left-most general cell. The resultant $b^N(x)$ can have at most $t+1$ coefficients, and we must ensure that as the $b^N(x)$ polynomial is growing toward the right that it is not corrupted by data moving toward the left from unused cells. At termination, cell $t+2$ is the first cell that will not contribute to the solution, and since the entire structure has been shifting left on each iteration, this cell corresponds to cell $3t+1$ in the initialized array. During initialization the least significant coefficient of $b^N(x)$, which resides in cell $2t$, must be followed by t zeros. If there are fewer than $3t$ general cells in the system, then the right-most cell must have all it's right-inputs set to zero and all the cells to the right of cell $2t$ must be initialized to zero.

The special left-cell also requires initialization. It has the identical initialization procedure as the general cells, and is part of the contiguous initialization shift-register. The first syndrome S_0, which enters the array last, is placed in the special left cell and represents r_j^N for $j=1$. The $r^O(x)$ polynomial is initialized to 1 (figure 2). All the general cells have their coefficient of $r^O(x)$ initialized to zero.

Every cell must perform $2t$ iterations of the algorithm presented in figure 2. With the completely pipelined 1-D system the left-most cell will finish first followed by the second cell, then the third all the way through to the $(t+1)$st cell. Each cell places its results into a static register, and when the $(t+1)$st cell is finished, these static registers are configured into a single bit-serial shift-register for outputting the results of the key equation processor.

Coalescing General Cell Pairs (error only)

For codes with rates greater than 1/2, the 50 percent cell utilization can be exploited to reduce the number of general cells by 50 percent. The significance of 1/2 rate codes is that for codes with rates which are greater than 1/2, there will never be syndromes available from a second block prior to the completion of the current block. The opportunity to fold the cell exists because within any pair of neighboring cells, only one is active at any particular time. By adding a small amount of additional control logic, the arithmetic section of a single cell can be shared between two adjacent general cells.

Consider the operation of the ith cell pair. (Remember that the right and left cells within this pair cannot be active simultaneously.) When the ith cell-pair's right-cell is active it receives inputs from the left-cell of the $(i+1)$st cell-pair and inputs from the left-cell of the ith cell-pair. Then when the left-cell of the ith cell-pair is active, it receives its inputs from the right-cell of the ith cell-pair and the right-cell of the $(i-1)$st cell-pair.

Two general cells can be replaced by the single folded cell of figure 6. Only a few changes are necessary. Four multiplexers are required to direct inputs and additional delays are needed for e, r_j^N, and r_j^O. The delays are increased because each of these signals that are generated by the special left cell must be seen by both the left-cell and right-cell of each cell-pair.

Erasures Too

Blahut demonstrated that the decoding of erasures can be performed using an identical algorithm to that of error-only decoding provided that modified syndromes [6] are used to initialize the algorithm [1]. This partitions the decoding algorithm into two distinct loops, an initialization loop that calculates the modified syndromes and a main recursion.

The initialization loop requires the number of erasures that exist for this data block $l=\mu$. The initialization loop calculates the modified syndrome polynomial $p^N(x)$

$$p^N(x) = xS(x)\prod_{k=1}^{\mu}(1-W_k x),$$

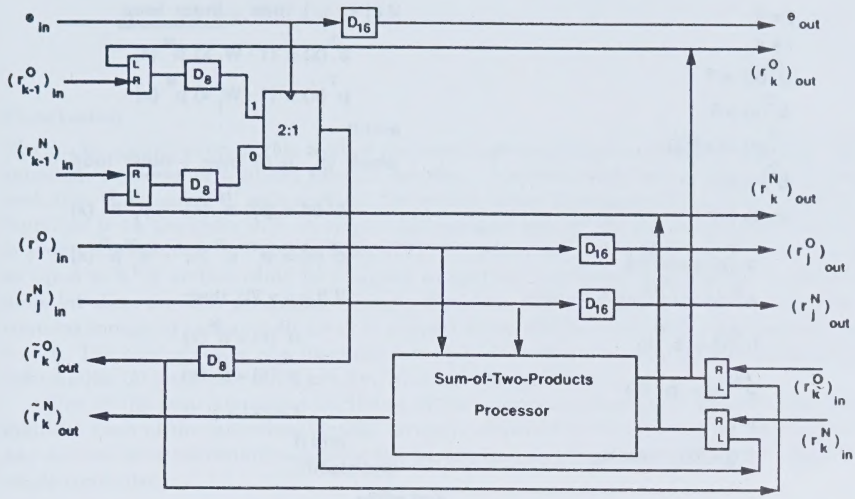


Figure 6. Folded 1-D General Cell (Error Only)

and the erasure locator polynomial that is used as the initialization of the connection polynomial

$$b^N(x) = \Upsilon(x) = \prod_{k=1}^{\mu}(1 - W_k x).$$

$p^N(x)$ is equivalent to $r^N(x)$ in the errors-only decoder. The W_k are the known erasure locations. With $p^N(x)$ and $b^N(x)$ initialized as above Eastman's array (figure 3) could be used to compute the final erasure and error locator polynomial

$$\Pi(x) = \Upsilon(x)\Lambda(x).$$

The number of stages required in the 2-D case is $2t - \mu$.

From a hardware perspective it is extremely desirable to imbed the initialization process within the normal computation structure so that the algorithm executes in a fixed number of steps independent of the number of erasures. Figure 7 shows an algorithm where the iterative generation of the erasure polynomial takes place within the main body of the program. There are two distinct loops within the main iteration. The inner loop establishes the initialization conditions for $b^N(x)$ and $p^N(x)$, and will execute μ times. The outer loop executes the remaining $2t - \mu$ times to complete the algorithm. In both loops the polynomials $b^N(x)$ and $p^N(x)$ are updated as a sum-of-two-products. Since we know when we are in each of these loops, we can use a single sum-of-two-products processor as the heart of an errata decoding cell. We control the inputs to this sum-of-products processor by knowing whether we are in the inner (erasure loading) or outer loop of figure 7.

The polynomial update on a coefficient-by-coefficient basis when executing the inner loop is

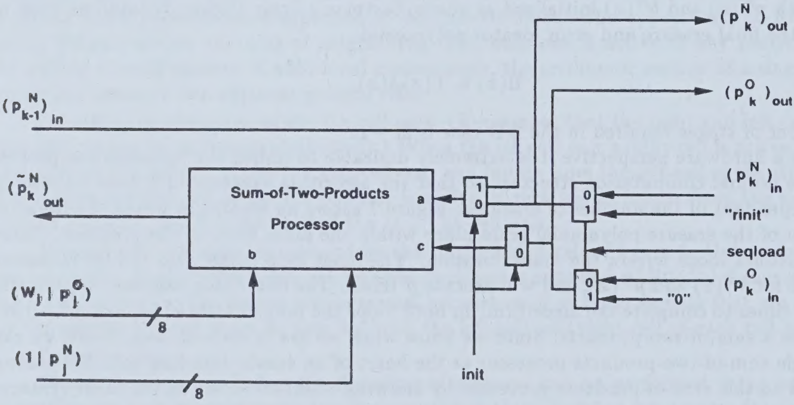
$$(\tilde{p}^N_k) = p^N_k - W_j p^N_{k-1}$$

Figure 8 shows the control of the sum-of-two-products processor. The sum-of-two products processor forms $ab + cd$, where a, b, c, and d are inputs. The special left cell must use the two 8-bit busses in two different manners, depending upon whether or not the system is executing the inner or outer loop. When the inner loop is executed, W_j must enter on the p^O_j port and

```
$j = 0$
$j = \mu$
$p^O(x) = 1$
$b^O(x) = 0$
$p^T(x) = xS(x)$
$b^T(x) = 1$
while (1)
    $b^O(x) = x\, b^O(x)$
    $p^O(x) = x\, p^O(x)$
    $b^N(x) = b^T(x)$
    $p^N(x) = p^T(x)$
    j++
    if (j > 2t)   break ;
    if ( $j <= \mu$ ) then ; inner loop
        $b^T(x) = (1 - W_j x)\, b^N(x)$
        $p^T(x) = (1 - W_j x)\, p^N(x)$
    end if
    elseif ($p_j^N$ != 0) then ; outer loop
        $b^T(x) = p_j^O\, b^N(x) - p_j^N\, b^O(x)$
        $p^T(x) = p_j^O\, p^N(x) - p_j^N\, p^O(x)$
        if ($j + \mu > 2l$) then
            $b^O(x) = b^N(x)$
            $p^O(x) = p^N(x)$
            $l = j - l + \mu$
        end if
    end elseif
end while
```

Figure 7. Key Equation Algorithm With Erasures

the value of 1 must enter the p_j^N port. When executing the inner loop, *init* is high and the circuit in figure 8 is seen to form the result $(\tilde{p}_k^N)$. When we enter the outer loop of figure 7, *init* is off and the general cell functions identically to the non-erasure cell presented in figure 6.



$(p_k^N)_{out}$
$(p_k^O)_{out}$
$(p_{k-1}^N)_{in}$
$(\tilde{p}_k^N)_{out}$
Sum-of-Two-Products
Processor
a
b
c
d
1
0
1
0
0
1
0
1
$(p_k^N)_{in}$
"rinit"
seqload
$(p_k^O)_{in}$
"0"
$(W_j \mid p_j^O)$
8
$(1 \mid p_j^N)$
8
init

Figure 8. Control of the Sum-of-Two-Products Processor

Conclusion

The major contribution of this work is the new implementation technique for solving the key equation, resulting in a purely systolic decoder. A system architecture has been developed such that the number of cells required for systolic array implementations for each of the five functions is on the order of t. A single chip contains each of the five major components and is programmable over the Galois fields $GF(2^m)$, where we have restricted m to be less than or equal to 8. A system could be designed to operate over larger fields simply by introducng more bit-slices for each processor. A single, 2 micron, CMOS chip (7900x9200 sq.-micron die) contains enough of each systolic array to support codes with an error correction capability up to $t = 16$. The chip operates at a bit-serial rate of 20 MHz, thereby supporting a communication information bit data rate of $20(k/n)(m/8)$ mega-bits per second.

One of the most attractive attributes of the system architecture is that the chip itself is systolic. Each of the individual systolic arrays is extended by arraying chips together, so that any desired error correction capabilty can be attained by using multiple systolic chips with a single controller.

References

[1] Blahut R.E., "Theory and Practice of Error Control Codes," Addison-Wesley, 1983.

[2] Brent R.P., Kung H.T., "Systolic VLSI Arrays for Linear-Time GCD Computations," VLSI '83, North-Holland, pp. 145-154.

[3] Demassieux N., Jutand F., Muller M., "A 10 MHz (255,223) Reed-Solomon Decoder," IEEE 1988 Custom Integrated Circuits Conference, 17.6.

[4] Eastman W.L., "Euclideanization of the Berlekamp-Massey Algorithm," 1988 Tactical Communications Conference.

[5] Eastman W.L., "Euclidean Decoders For BCH Codes," RADC-TR-88-44 Final Technical Report, April 1988.

[6] Forney, G.D., "On Decoding BCH Codes," IEEE Trans. on Inf. Theory, IT-17 (1971), pp. 464-466.

[7] Liu K.Y., "Architecture for VLSI Design of Reed-Solomon Decoders," IEEE Trans Computers, vol. C-33, no. 2, Feb. 1984, pp. 178-189.

[8] Shao H.M., et al., "A VLSI Design of a Pipeline Reed-Solomon Decoder," IEEE Trans Computers, vol. C-34, no. 5, May 1985, pp. 393-403.

GENERALIZED DE BRUIJN SEQUENCES

ROBERT J. MCELIECE
LADA POPOVIĆ
California Institute of Technology
Department of Electrical Engineering
MC 116-81, Pasadena, CA 91125

Abstract : A (q, ν) de Bruijn sequence is a periodic sequence of letters from a q-ary alphabet such that any given ν-tuple appears exactly once as a " window" in a period. These sequences are well known and have been used in a variety of applications. We introduce the family of *Generalized de Bruijn Sequences*, which involves several sequences in parallel and an irregularly shaped window, plus the requirement that every possible window content should appear n times per period. The existence of such sequences is proved by construction, and a formula for their number is derived. The classical de Bruijn sequences can then be regarded as a special case of this new family.

1. INTRODUCTION

A q-ary de Bruijn sequence $dB(q,\nu)$ is a periodic sequence of symbols from a q-letter alphabet such that each of the q^ν different ν-tuples of letters appears exactly once as a window in each period. An example for $q = 2$, $\nu = 3$, appears in Fig. 1 . These sequences have been known for a long time, but their extensive study in the engineering literature began after it was discovered that various applications can be made in communications, computer science, cryptography etc. A good review of the area can be found in [1], [2].

The problem has a natural generalization to two dimensions: the 2-dimensional, $r \times v$-periodic sequences in which each possible content of an $n \times m$ window appears exactly once per period. For example, in Fig. 2 the 16 binary 2×2 matrices all appear in a 4×4-periodic array. The two-dimensional de Bruijn sequences have found applications in optics, communications, and other areas. In a recent paper ([3]) a list of references on the subject can be found.

This paper introduces a class of de Bruijn-type objects which occupy a middle ground between one and two-dimensional de Bruijn sequences. We adopt the name Generalized (1-dimensional) de Bruijn sequences, or *GdB* sequences. In order to construct them, we first consider two special cases: Type 1 and Type 2 sequences.

For a Type 1 (GdB_1) sequence, a q-letter alphabet and positive integers k, $\nu \geq k$ are given. A $k \times q^\nu$ matrix of letters from the alphabet is periodically and indefinitely extended to the left and right, thus forming a periodic array. An irregularly shaped window (Fig. 3) of height k and area ν is superimposed on this array. As the window slides step by step to the right the requirement is that each possible window content should appear exactly once in a period. An example of such a sequence appears in Fig. 4 .

We give a construction for GdB_1 sequences in Section 2, along with a formula for the number of distinct GdB_1 sequences for a given set of parameters.

In Section 3, another possible generalization of 1-dimensional de Bruijn sequences is proposed: a periodic sequence of letters with the property that every possible ν-tuple appears exactly n times per period, and that each appearance of a particular ν-tuple has a different position modulo n in a period. Such is the sequence in Fig. 7 : the four different shapes, each of which encloses $\nu = 2$ letters, correspond to the positions modulo $n = 4$. We call these Type 2 *GdB* sequences (GdB_2). Again, we give a construction to generate such sequences and derive a formula for their number.

Finaly, Section 4 introduces the Generalized de Bruijn sequences, which are just the logical synthesis of the Type 1 and Type 2 sequences. The construction and the formula for their number are also given.

The available space has compelled us to dispense with somewhat cumbersome notation and large number of technical lemmas which are necessary for the precise, formal derivation of the results presented here. The interested reader will be able to find them all in [5]. The present paper, however, is self-contained and we hope that the informal approach addopted here will make it in fact more accessible than [5].

2. TYPE 1 GdB SEQUENCES

The construction for GdB_1 sequences that we propose is a natural extension of the construction for 1-dimensional de Bruijn sequences that can be found in [1]. In addition to the parameters q, k, ν mentioned in Section 1 (see Fig. 3), we need to specify positive integers $(\nu_1, \nu_2, ..., \nu_k)$, whose sum equals ν, which determine how many letters of a particular row in the array are visible through the window (without loss of generality we may assume $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_k \geq 1$). We denote the set of distinct GdB_1 sequences with this set of parameters

$$GdB_1(q, k, \nu; \nu_1, \nu_2, \cdots, \nu_k)$$

The sequence in Fig. 4 thus has parameters $(2, 2, 3; 2, 1)$.

Next, we define a *Generalized de Bruijn* (GB) graph for the same set of parameters to be the direct product of the k ordinary de Bruijn graphs $B(q, \nu_i - 1)$, for $i = 1, 2, \cdots, k$. The vertices of $B(q, \nu - 1)$ are all $(\nu - 1)$-tuples over a q-letter alphabet; there is an edge from $\mathbf{x}$ to $\mathbf{y}$ iff $y_2 = x_1, y_3 = x_2, \cdots, y_{\nu-1} = y_{\nu-2}$, and the edge is labelled by the ν - tuple $(y_1, y_2, \cdots, y_{\nu-1}, x_{\nu-1})$. Fig. 5 shows the Generalized de Bruijn graph which is the direct product of the de Bruijn graphs $B(2, 0)$ and $B(2, 1)$.

If edges and vertices of the product graph are labelled in the usual way by pairs of labels from the original graphs, this construction yields a natural 1-1 labelling of the edges of the GB graph by the possible contents of the window. Thus edge a follows edge b iff the window content that labels the edge a can be obtained by pushing the contents of the label of b one step to the right and inserting a new column of k letters on the left side. The intermediate vertex is labelled by the overlap (Fig. 6). It is not difficult to show that this labelling establishes a 1-1 correspondence between the Euler circuits of the graph and all GdB sequences for a given set of parameters. Traversing the edges of the graph $GB(2; 1, 0)$ of Fig. 5 in the order indicated in Fig. 6 one obtains (from right to left) the sequence of Fig. 4 (the window contents correspond to the edge labels).

It is possible to determine the number of Euler circuits in a GB graph and thus also the number of GdB_1 sequences. To do so, we observe that any GB graph may be built from a 1-state graph by an alternating sequence of the following two operations, each of which acts on a graph G and gives a new graph G':

- the "doubling" operation: for some l, replace each edge in G by l "parallel" edges (between the same states, in the same direction);

- the "higher edge" operation: let the vertices of G' be the edges of G , and let there be an edge from a to b in G' iff edge b follows edge a in G.

To see that this is indeed the case, imagine that we have a window like the one in Fig. 3. We have hitherto specified its shape by the row lengths $(\nu_1, \nu_2, \cdots, \nu_k)$, but we could have given the column lengths $(\eta_1, \eta_2, \cdots, \eta_m)$ instead. We start with a 1-state graph with q^{η_m} edges labelled by all possible contents of the 1-column subwindow (η_m). Next we use the higher edge operation, followed by the doubling operation with parameter l equal to $q^{(\eta_{m-1}-\eta_m)}$. The edges of the new graph can now be labelled by all contents of the subwindow (η_{m-1}, η_m). Continuing in this way, we eventually build up the graph which corresponds to the whole window.

Generalized de Bruijn graphs are regular, which means that the number of outgoing or incoming edges is the same for every vertex. Theorems 3 and 4 in [4] give the relation between the number of Euler circuits in G and G' under each of the above two operations if G is regular . Using that, we obtain the formula

$$\mid GdB_1(q, k, \nu; \nu_1, \nu_2, \cdots \nu_k) \mid = q^{-\nu}((q^k)!)^{q^{\nu-k}}$$

for the number of GdB_1 sequences. Thus the number of sequences depends on the area ν of the window, not on its actual shape.

3. TYPE 2 GdB SEQUENCES

A (q, ν, n), Type 2 Generalized de Bruijn sequence is an nq^ν-periodic 1-dimensional array of letters from a q-ary alphabet with the property that each ν-tuple appears exactly n times as a window in a period, and that each appearance has a different position modulo n in the array. In other words, if a window of length ν is placed succesively at positions $i, i+n, \cdots, i+(q^\nu-1)n$ in the array, it shoud display distinct ν-tuples. A GdB_2 sequence with parameters $(2, 2, 4)$ appears in Fig. 7.

In order to construct such a sequence, we "unfold" the de Bruijn graph $B(q, \nu-1)$ into a cyclic *trellis* of length n. This is the same as forming the direct product of $B(q, \nu-1)$ with the cycle graph on n vertices C_n. It can be shown ([5]) that the sequences of Type 2 with parameters (q, ν, n) correspond $1-1$ to the Euler circuits of this graph. To get a feeling for this correspondence, the reader may try to construct a sequence like the one in Fig 7. by outlining an Euler circuit in $C_4 \times B(2, 1)$.

We make use of two lemmas (proved in [5]):

Lemma 1: $B(q, \nu) = (B(q, \nu-1))'$, where the prime denotes the higher-edge operation.

Lemma 2: For any graphs G_1, G_2

$$(G_1 \times G_2)' = G_1' \times G_2'$$

The chain of equalities

$$B(q,\nu-1) \times C_n = B'(q,\nu-2) \times C_n = B'(q,\nu-2) \times C_n' = (B(q,\nu-2) \times C_n)'$$

reveals that the graphs of higher order can be obtained using the higher edge operation. The corresponding relation in [4] for the number of Euler circuits then yields the formula

$$\mid GdB_2(q,\nu,n) \mid = q^{-\nu}(q!)^{nq^{\nu-1}}$$

for the number of sequences of Type 2.

4. GENERALIZED DE BRUIJN SEQUENCES

The goal is now to obtain a generalization of both Type 1 and Type 2 sequences. We do this by modifying the definition of Type 1 sequences in the similar way as the definition of (ordinary) de Bruijn sequences was modified to obtain Type 2 sequences. First, we specify a window like the one in Fig. 3, of height k and area ν, displaying ν_i letters in row i for $i = 1, 2, \cdots, k$.Then an additional parameter n is introduced , which will play the same role as in the Type 2 sequences. A *Generalized de Bruijn Sequence* consists of letters from a q-ary alphabet arranged in a $k \times nq^{\nu}$ matrix, extended periodicaly to the left and right, with the property that, starting at any position i, the window displays distinct contents at positions $i, i+n, i+2n, \cdots, i+(q^{\nu-1})n$. When $n = 1$ this is just the definition of Type 1 sequences. The Type 2 sequences can be obtained as well – setting $k = 1$.

The construction for new sequences can be deduced from the two previos cases. The graph which represents them is a product of the Cycle graph C_n with the Generalized de Bruijn graph corresponding to the window parameters. The methods developed in Sections 2 and 3 yield the formula for the number of GdB sequences

$$\mid GdB(q,k,\nu,n;\nu_1,\cdots,\nu_k) \mid = q^{-\nu}((q^k)!)^{nq^{\nu-k}}$$

Of course, this formula contains the previous two as special cases: $n = 1$ gives the Type 1 sequences, $k = 1$ those of Type 2. If both $k = 1$ and $n = 1$ we have the ordinary de Bruijn sequences and the above expression reduces to the well-known formula

$$\mid dB(q,\nu) \mid = q^{-\nu}(q!)^{q^{\nu-1}}$$

REFERENCES

[1] S. W. Golomb, *Shift Register Sequences.* Laguna Hills, CA: Aegean Park Press, 1982.

[2] H. M. Fredericksen, " A survey of full length nonlinear shift register cycle algorithms", *SIAM Rev.* , vol. 24, pp. 195-221, Apr. 1982

[3] T. Etzion, " Constructions for perfect maps and pseudorandom arrays", *IEEE Trans. Inform. Theory*, vol. 34, pp. 1308-1316, Sept. 1988

[4] T. van Aardenne-Ehrenfest and N. G. de Bruijn, "Circuits and trees in ordered linear graphs", *Simon Stevin*, vol. 28, pp. 203-217, 1951.

[5] L. Popović, *PhD Thesis*; California Institute of Technology. *In preparation.*

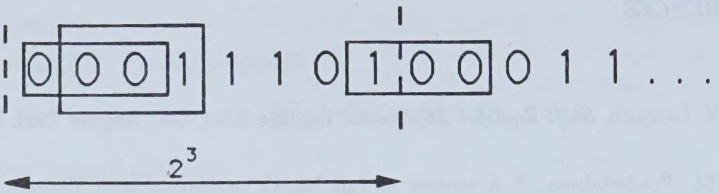

0 0 0 1 1 1 0 1 0 0 0 1 1 . . .
2³

Fig. 1

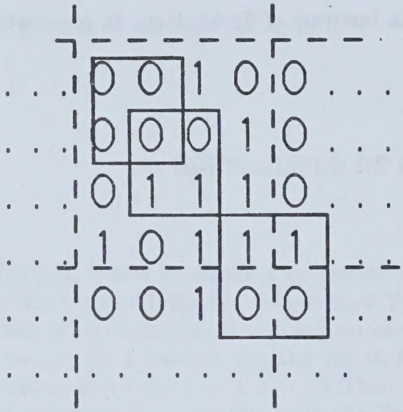

. . . 0 0 1 0 0 . . .
. . . 0 0 0 1 0 . . .
. . . 0 1 1 1 0 . . .
. . . 1 0 1 1 1 . . .
. . . 0 0 1 0 0 . . .
. . . .

Fig. 2

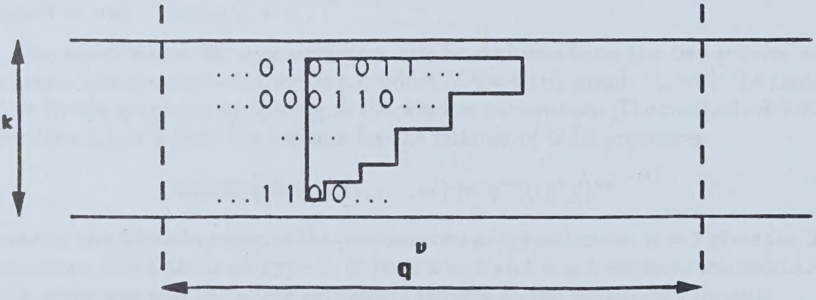

. . . 0 1 0 1 0 1 1 . . .
. . . 0 0 0 1 1 0 . . .
. . . 1 1 0 0 . . .
k
q^ν

Fig. 3

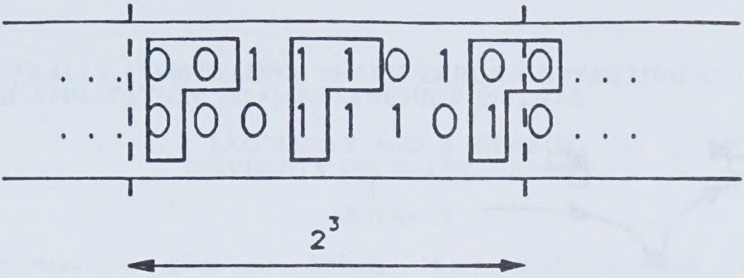
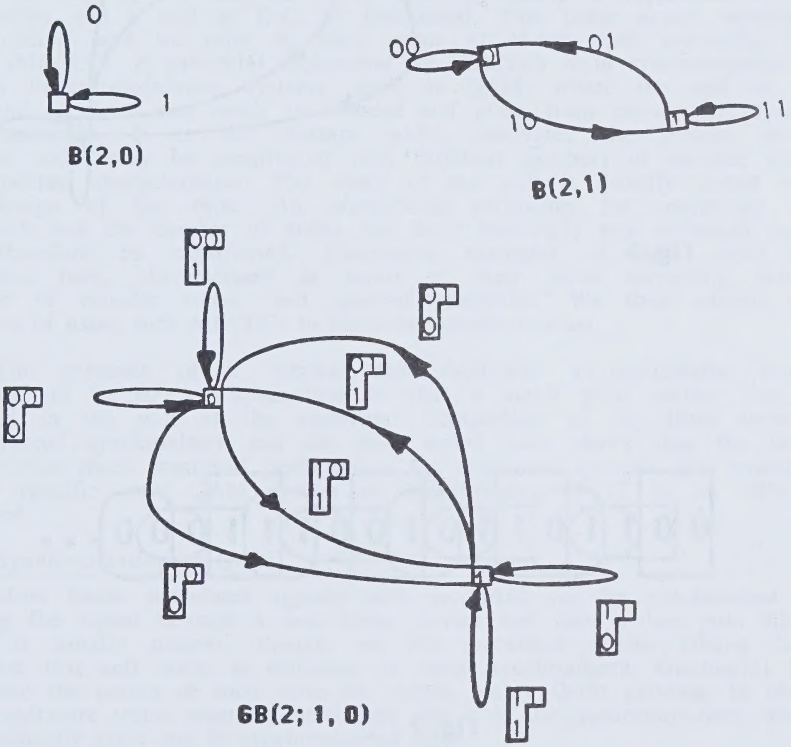

. . . 0 0 1 1 1 0 1 0 0 . . .
. . . 0 0 0 1 1 1 0 1 0 . . .
2^3

Fig. 4

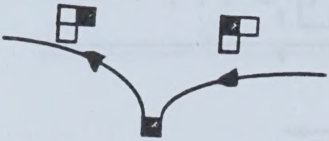

0
1
00
01
10
11

B(2,0)

B(2,1)

GB(2; 1, 0)

Fig. 5



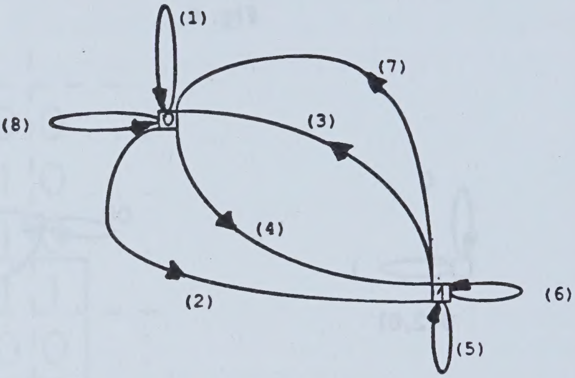

(1)
(7)
(8)
0
(3)
(4)
(2)
1
(6)
(5)

Fig. 6

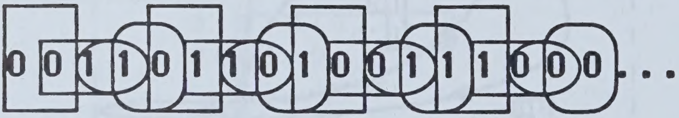

0 0 1 1 0 1 1 0 1 0 0 1 1 1 0 0 0 . . .

Fig. 7

SPECTRALLY CONSTRAINED M-ARY ERROR CORRECTING CODES WITH APPLICATION TO QAM SYNCHRONISATION.

J.J.O'REILLY AND A. KOKKOS
UNIVERSITY OF WALES, BANGOR.

ABSTRACT

Consideration is given to devising M-ary error correcting codes to incorporate also line coding features providing a D.C. null. This results in a null at the carrier frequency when applied to QAM transmission which, it is shown, can offer potential benefits for synchronisation for M-ary systems.

1. Introduction.

In this paper we address the issue of designing M-ary error correcting codes, in such a way that the power spectral density is suppressed at low frequencies and a null at D.C. is introduced. This latter aspect represents 'line coding' and we refer to these codes as M-ary error correcting line codes (MECLC). A particular application for MECLCs is in synchronisation of certain linear modulation systems, such as QAM, where the null in the baseband spectrum can result in reduced self noise from carrier and symbol synchronisation. It is this feature which motivates the present study. Various codes may be constructed with different numbers of encoder states and spectral characteristics. The width of the null is directly linked with the design of the code. An algorithmic procedure for optimising the nullwidth and the number of states has been devised[1] and optimised codes can therefore be constructed. Illustrative examples of such codes are presented here, characterized in terms of their error correcting power, number of encoder states, and spectral properties. We then address the question of using such MECLCs to facilitate synchronisation.

The presence of a spectral null facilitates a particularly simple approach to carrier synchronisation in that a small pilot carrier can be inserted in the null of the spectrum. Comparison of the jitter between conventional synchronisers and the pilot coded ones shows that the latter can provide much improved performance for multilevel systems. For example, for a specific octal QAM system an improvement of 12 to 35 dB's is achieved.

2. Synchronisation of QAM.

Most linear modulated signals such as QAM can be synchronised by passing the signal through a non linear device and then a low pass filter. Jitter is usually present, though, on the recovered carrier arising from the fact that self noise is dominant in these synchronisers. Gardner[2] has examined the nature of such noise for simple binary QAM systems. In phase and quadrature terms must be considered and it is the quadrature term which predominantly gives rise to synchronisation jitter.

A four phase synchronous data signal can be represented as[2]:

$$v_i(t) = x(t)\cos(\omega_i t+\theta_i) - y(t)\sin(\omega_i t+\theta_i) \qquad (1)$$

where x(t) and y(t) are the in-phase and quadrature components of modulation, ω_i is the receiver carrier frequency and θ_i is an arbitrary fixed phase. A carrier at frequency $4\omega_i$ can be recovered by applying the signal of (1) to a fourth order nonlinearity. If x and y have zero D.C. level and are independent of one another, then the amplitude of the recovered carrier is given by:

$$A = \frac{1}{8}\mathrm{avg}(x^4+y^4-6x^2y^2) \qquad (2)$$

where the notation avg( ) denotes averaging over data and over time. The in-phase self noise envelope can then be identified as:

$$a(t) = \frac{1}{8}(x^4+y^4-6x^2y^2) - A \qquad (3)$$

and the quadrature self noise envelope as:

$$b(t) = \frac{1}{2}\, xy(y^2-x^2) \qquad (4)$$

Let x(t) and y(t) be representable as:

$$x(t) = \sum_n c_n\, g(t-nT) \quad , \qquad y(t) = \sum_n d_n\, g(t-nT-kT/2)$$

where

$$c_n,\ d_n = \pm 1, \pm 3, \ldots, \pm(M-1)$$

and g(t) is the signaling pulse waveform, strictly time limited to the symbol interval T.

The envelope functions a(t) and b(t) can be regarded as random synchronous pulse trains. A method for determining the spectrum of such pulse train is described by Bennett and Davey[5]; Using this method we have calculated the self noise spectra for binary and multi-level QAM numerically.

For binary signaling the resulting quadrature spectral component of the self noise exhibits a null at D.C. and this gives rise to improved performance of the synchroniser. In multilevel QAM systems though, this spectral property does not hold in general; we have examined specifically certain multilevel QAM systems and found that the resulting self noise spectrum does not exhibit a null at D.C. This may be appreciated from Figures 1 and 2 which represent the self noise in-phase and quadrature spectra for binary and octal QAM respectively. We note that for the octal case there is no D.C. null; This will degrade the performance of the synchroniser and may result in more complex solutions being required. Consequently we now turn our attention to the possibility of introducing coding which in addition to providing for error control will also facilitate synchronisation.

3. M-ary codes with spectral nulls.

For digital transmission the processes of error control coding and line coding are generally viewed as distinct and are implemented separately. The first makes use of redundancy to combat digital errors induced by noise and interference whilst the second is concerned with introducing redundancy to control other properties such as the power spectral density of the transmitted signal. Where both operations are required they are traditionally implemented in cascade.

Recently there has been some interest in devising codes which achieve these two sets of objectives in a single, cooperative realization. The main emphasis to date has been on binary ECLCs but for certain practical channels multilevel signals can be particularly appropriate. This is perhaps especially the case when considering coding in combination with modulation processes. For the cascade operation there are efficiency limitations a conventional line code introducing a substantial rate reduction without a concomitant benefit in error control power. This means that the efficiency of the cascade system can never exceed that of the particular line code adopted, with additional rate reduction arising from the error control code. Efficiency limitations associated with the relative ratio reduction incurred as a result of imposing line code sequence restrictions no longer exist in the combined system: the rate of ECLCs is directly linked with the rate of the parent Error Correcting Code (ECC).

The basic approach we adopt in designing ECLCs is to form a line code with codewords available from a known ECC of appropriate radix. There are, of course, many ECLCs that can be so constructed and there are trade-offs to be obtained between various factors - such as the codeword length, the number of encoder states, and the size of the code look up table involved - and this will influence also the spectral characteristics of the resulting signal. Accordingly we have devised an algorithmic procedure for designing ECLCs, multilevel or binary, and a number of MECLCs are illustrated by way of their code power spectral densities, calculated using the method of Cariolaro Tronca[3]. One of the aims when optimising MECLCs is to obtain power spectra with a null at D.C. and low response near the D.C. region. The frequency at which the power spectrum attains half of its maximum value is an indication of how wide the null at D.C. is, required to be as wide as possible for some applications. Illustrative MECLCs code spectra are presented in Figure 3. The code design algorithm is presented in detail in [1] and is thus only summarised here in the appendix. Optimised codes can then be constructed and a study of a range of binary and multilevel cases, summarised in Figures 4, 5 and 6, reveals the following:

1. For binary ECLCs the codes that have the minimum number of states do not have the widest nullwidth. Hence for the binary case the codes that provide the widest nullwidth are not the simplest possible ones.

2. For the quaternary case the codes that provide the widest nullwidth do give the minimum number of encoder states.

3. For the octal case we have studied certain '14 binary 7 octal' single ECLCs. Three specific codes were constructed: a code with 44 states, a code with 49 states and one with 50 states. Of these, the code with the widest nullwidth was that with 44 states (Figure 3) while the code with 49 states was found to have the narrowest nullwidth.

Construction and study of multilevel codes though, is rather complicated when the number of signal levels is large and in order to generalise results for the octal case further work will be required. For the purpose of the present analysis the 50 state code was used while the illustrative binary case example selected relates to a 16 state '4 binary 15 binary' triple ECLC. With this preparation we are now in a position to asses the effectiveness of MECLCs in facilitating synchronisation of linear QAM-based modulation schemes.

4. Performance consideration of coded and uncoded synchronisation systems.

As mentioned previously, the synchronisation of linear modulation systems can be effected by passing the signal through a suitable non linear device and then a low pass filter. For the coded systems developed here, though, the simple expedience of inserting a low level pilot tone in the spectral null provides an alternative which deserves attention since synchronisation can then be achieved directly using a Phase Locked Loop (PLL) or a tuned circuit.

For the conventional PLL synchroniser the phase variance caused by self-noise is the integral:

$$\sigma_\theta^2 = \frac{1}{A^2}\int_{-\infty}^{\infty} | H(j2\pi f) |^2 S_b(f)\, df$$

where $H(j2\pi f)$ is the closed loop phase transfer function and $S_b(f)$ is the quadrature noise spectrum. For the pilot tone systems there is a similar expression for the phase variance, but in this case $S_b(f)$ represents the coded or uncoded power spectrum density rather than the self-noise spectrum, and A is the amplitude of the pilot tone which can be of any arbitrary value rather than having the fixed value of eqn 2 for the conventional synchroniser.

We have accordingly examined the synchroniser jitter for binary QAM and octal QAM, considering three cases : a) a conventional QAM synchroniser b) coded pilot tone synchroniser and c) uncoded pilot tone synchroniser. Carrier synchronisation jitter results are presented in figures 7 to 10, from which we observe the following:

1. For the binary case the conventional synchroniser gives better performance than both the coded and uncoded pilot tone cases. This arises because for binary QAM the non linear processing produces a null at D.C. in the quadrature self noise spectrum. Results have been produced for two cases a) when an ideal filter is user for loop tracking and b) when a simple second order tuned circuit is used for loop tracking. These represent respectively best and realistically worst cases; the performance of a specific practical synchroniser will be somewhere between these two extremes. The performance of the conventional synchroniser is 35 dB better than the coded case and 50 dB better than the uncoded case when ideal filtering is considered, while when a tuned circuit is used the conventional synchroniser gives 25 dB better performance than the coded case and 35 dB better than the uncoded case.

2. For the octal case, though, these observations no longer hold. The coded system is superior now to the conventional synchroniser, offering 35 dB better performance when ideal filtering is used and 12 dB for the tuned circuit. The uncoded pilot tone system has the worst performance in all cases demonstrating convincingly the capability of ECLCs for providing carrier synchroniser self noise reduction.

5. Conclusion.

The possibility of modifying M-ary error control codes in such a way as to improve and/or simplify carrier synchronisation in linear modulation schemes has been examined. The codes belong to a new class of error correcting codes with defined spectra, designated as M-ary Error Correcting

Line Codes (MECLCs). Such codes provide error control as well as suppressed low frequency spectrum and a null at D.C. An algorithm that optimises the complexity of the encoder and the nullwidth has been devised, providing a basis for optimised code design. When such codes are used in addition to the implicit error control characteristics inherited from the particular error control code improved and simplified synchronisation may be achieved for M-ary QAM digital signaling. A number of illustrative examples have been presented on the basis of which we conclude that these coding techniques can offer increased flexibility in the design and realisation of synchronisation systems for linear digital modulation systems.

References.

1. Kokkos,A. and O'Reilly,J.J. 'A strategy for designing multilevel codes with defined spectrum and error control properties'. Institute of Mathematics and its Applications, Conf. Coding and Cryptography, Cirencester, UK, December 1989 (to be presented).

2. Gardner,F.M. 'Self noise in synchronisers'. IEEE Trans. Commun. Vol Com-28, No 8, August 1980 pp 1159 - 1163.

3. Herro,M.A. and Deng,R.H. 'Error correcting dc free binary transmission codes for fiber optic digital communications', Proc. Conf. Inf. Sci. and Systems, Baltimore, March '87 pp 559 - 564.

4. Popplewell,A and O'Reilly,J.J 'Performance aspects of error correcting line codes'. Proc. Second IEE National Conference in Telecommunications, York, April 1989 pp 47 - 52.

5. Bennett,W.R. and Davey,J.R, Data Transmission, New York: McGraw-Hill, 1965, ch.19.

Figure 1.

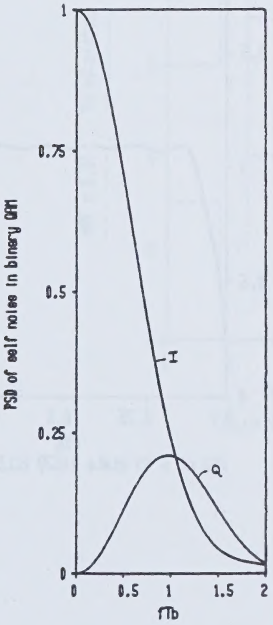


Figure 2.

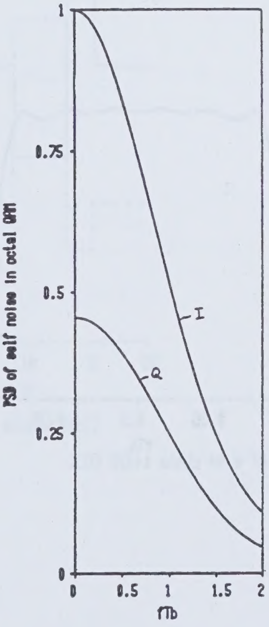
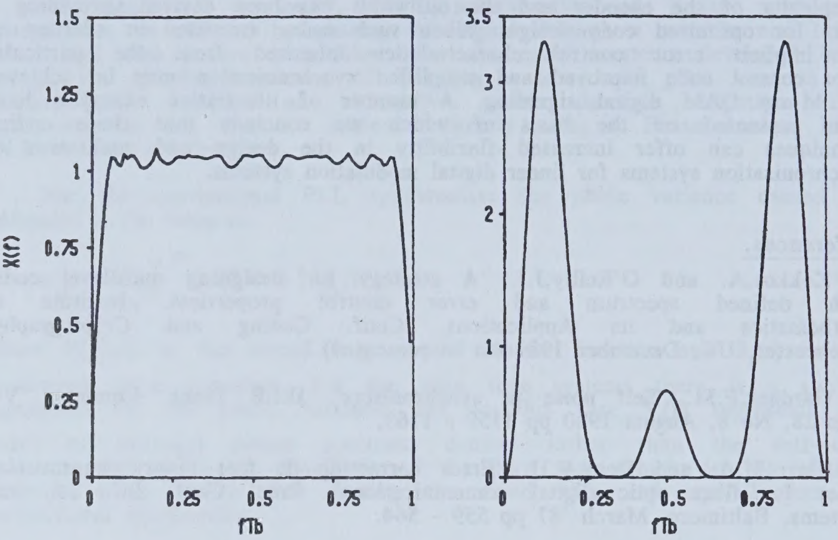
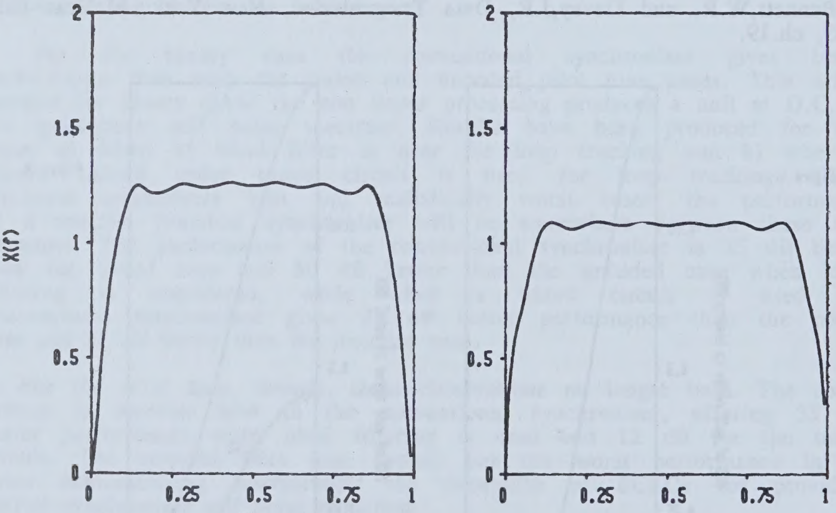


Figure 3.



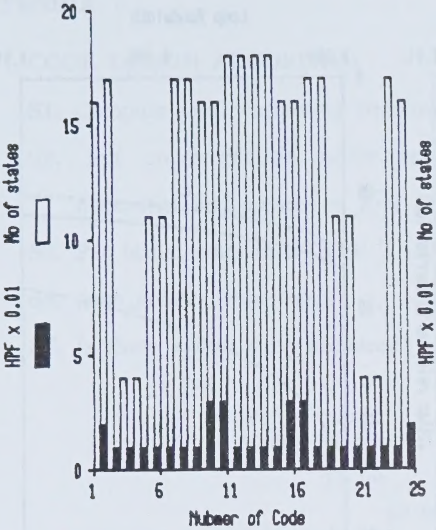
PSD of 4B15B triple ECLC

PSD of 1B3Q single ECLC



PSD of a 44 state 14B70 ECLC

PSD of a 49 state 14B70 ECLC



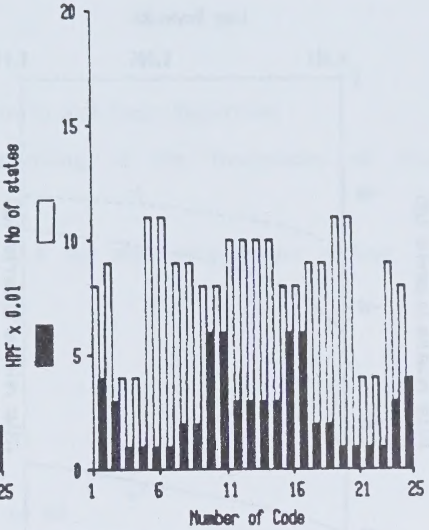


Figure 4. (4B15B triple ECLC)

Figure 5. (3B7B single ECLC)

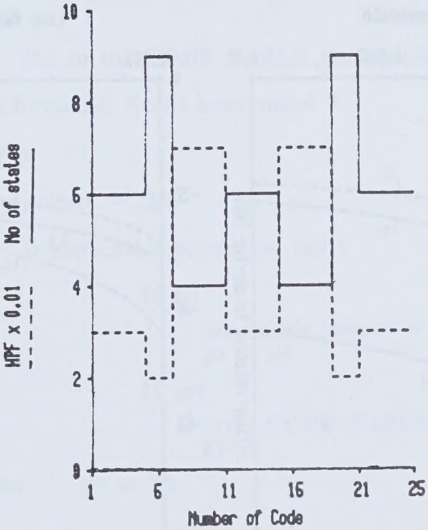


Figure 6. (1B30 single ECLC)

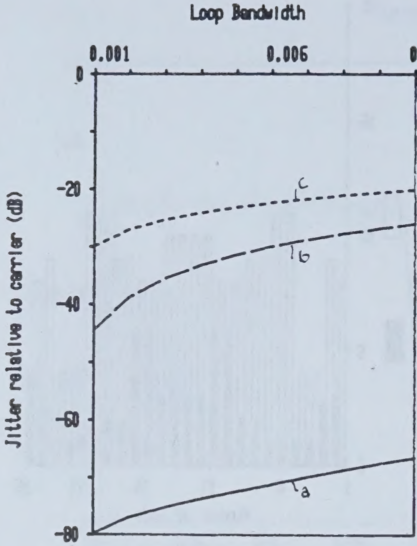


Figure 7.

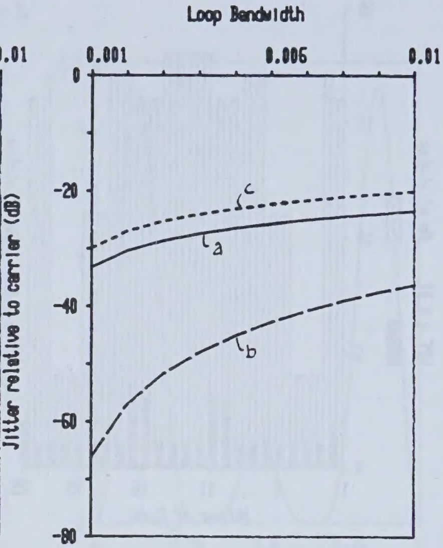


Figure 8.

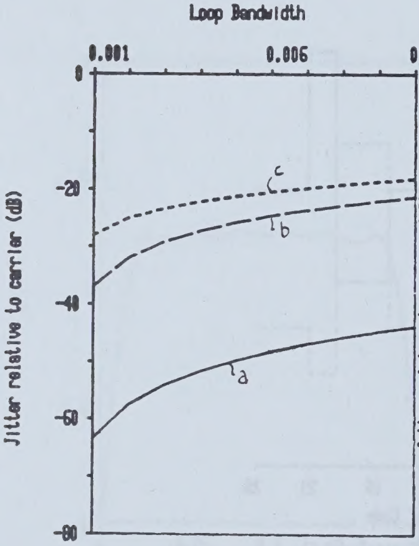


Figure 9.

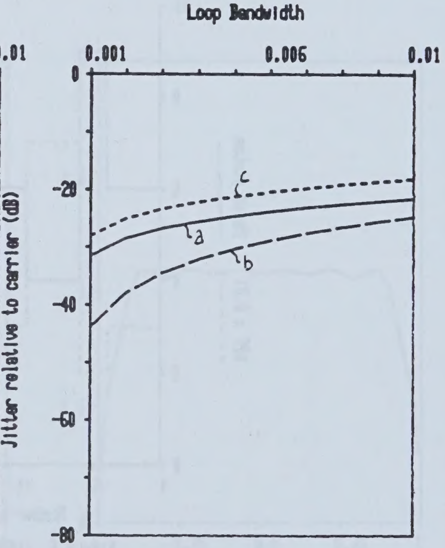


Figure 10.

In all Figures a:conventional b:coded pilot c:uncoded pilot

APPENDIX

CODE DESIGN ALGORITHM

```
S1. Compute error correcting codewords and their disparities.
S2. Put codewords in order according to the frequencies of their
    disparities and pointer = first
S3. Put initial state in STATE TABLE and state_table_pointer = first
S4. temp_pointer = pointer
S5. Is there a new state produced?
        If yes
            pointer = next
            Is pointer = last ?
                If no
                    go to S5
                If yes
                    pointer = temp_pointer
                    put new state in STATE TABLE
                    go to S6
        If no
S6.         Put in the CODE TABLE the codeword that pointer shows.
S7. Have all information words been coded ?
        If yes
            pointer = first
            Is state_table_pointer = last ?
                If no
                    state_table_pointer = next
                    go to S4
                If yes
                    Print CODE TABLE
                    STOP
        If no   go to S4
```

The Queue In A Statistical Multiplexer, As Seen By An Arriving Packet

Frank M. Brochin John B. Thomas

Princeton University
Department of Electrical Engineering
Princeton, NJ 08544

Abstract

Devising networks that could efficiently "integrate" both voice and data communications promises several technical and economical benefits. With the emergence of the Integrated Services Digital Networks and the introduction of Statistical Multiplexing, a large amount of research has been devoted to the study of the delay performances of Statistical Multiplexers for mixed voice/data traffics. In this paper, we present a method for the determination of the queue length probability density function in a Statistical Multiplexer, ***as it is seen at the instant of arrival by a packet joining the queue.*** The model that we use for the description of the speech dynamics is a ***three-state Markov chain with two different types of silence***, yielding exponentially distributed active periods and hyper-exponentially distributed silence periods.

1 Introduction

Today's public telecommunication networks are composed of disjoint networks for voice and data applications. Each network has its own interfaces, transport systems, switching mechanisms and is therefore not suitable for other types of applications [10]. However, devising networks that could efficiently "integrate" both voice and data communications promises several technical and economical benefits. This is why a large amount of research has been devoted to the emerging Integrated Services Digital Networks. The most promising method for integration seems to be Fast Packet Switching, which is based on fiber optic links and simplified protocols [5,7]. Because some traffic streams are very bursty and would suffer from any other mixing principle, some proposals for this scheme involve the use of Statistical Multiplexing [5] at the entrance to the network. In a Statistical Multiplexer, voice, data and video packets are fed to a common queue by a finite number of sources and are removed from this queue on a First-Come First-Served basis for transmission over a single fast channel handling packets from all sources. Because of the real-time constraint on the voice transmission [13], it is necessary to compute the probability distribution function of the delay that packets experience in the queue of the multiplexer in order to evaluate the network performances [1,6,12,4]. To date, most approaches attempting to evaluate this distribution have been restricted to the study of the equilibrium probabilities for the queue length in the Statistical Multiplexer. These probabilities may be interpreted as the portion of time that the queue length is in each possible state [8]. However, it is well known [8] that these probabilities are generally not the same as the probabilities for an arriving customer to find the queue length in that state. It is therefore important to compute the probability distribution function for the queue length as it is seen by an arriving customer. In this paper, we give a method for such a computation in a mixed voice/data traffic case. The method that we propose uses a three-state Markov chain model for the description of the speech dynamics on each line, yielding exponentially distributed active periods and hyper-exponentially distributed silence periods.

2 The Statistical Multiplexer

We model the Statistical Multiplexer as a single server queue with a first-in first-out (FIFO) service discipline. We assume that the service times are regularly and deterministically spaced in time, the inter-service time being denoted by D (in seconds). We also assume that the waiting room in the system is of infinite dimension. A representation of the model of the Statistical Multiplexer is given in Figure 1.

As for the arrival traffic, we consider a finite number N of voice sources being multiplexed onto the common transmission link, along with an indefinite number of heterogeneous data sources. The model chosen for the representation of the speech dynamics is represented in Figure 2. Successive active and idle periods form an alternating renewal process and because of the memoryless property of the exponential distribution, the number of packets generated during each active period is assumed to be geometrically distributed. We can therefore approximate the length of each active period by $L\,T$, where L is geometrically distributed with parameter p and T is the packetization time. The data arrival traffic results from the superposition of all the data streams and is modeled on a single line. We shall use both the Poisson and Packet Train [9] models as representations for the data traffic.

3 The method of computation

In a recent paper [2], we presented a method of computation for the probability density function of the queue length in the Statistical Multiplexer, as it appears at the instant of arrival to a packet joining the queue. Our study was restricted to an arrival traffic consisting only of voice packets only. We extend here this method to the mixed voice/data case.

3.1 The embedded Markov chain

We assume here that the aggregate arrival process to our system can be represented by a renewal process. We shall show later in the paper how to approximate an arrival point process by a renewal process. We denote by q_n the number of packets found in the system immediately prior to the arrival of packet n, and by s_{n+1} the number of packets served between the arrival of packets n and $n+1$. It is clear that the following relation holds :

$$q_{n+1} = q_n + 1 - s_{n+1}$$

and since we chose to model the arrival process as a renewal process, the sequence q_n forms a discrete-state Markov chain.

3.2 The transition probabilities

We can define the transition probabilities for this Markov chain as

$$p_{ij} = P[q_{n+1} = j | q_n = i]$$

so that p_{ij} is the probability that exactly $i-j+1$ packets are removed from the queue during the interarrival time of packets n and $n+1$.

These probabilities [2] can be written as functions of the probabilities P_i of having exactly i service instants during the inter-arrival time. We obtain

$$\begin{array}{lll}
\text{For } j \geq i+2 & p_{ij} = 0 & \\
\text{For } j \leq i+1 & i = 0 & p_{01} = P_0 + P_1 \\
 & & p_{00} = 1 - P_0 - P_1 \\
 & i \geq 1 & p_{ij} = P_{i+1-j} \\
 & & p_{i0} = 1 - \sum_{n=0}^{i} P_n
\end{array}$$

The P_k's can also [2] all be written in the same form as

$$P_k = \int_{(k-1)\,D}^{k\,D} \frac{A(z+D) - A(z \vee 0)}{D}\, dz$$

where $(z \vee 0)$ denotes the larger of 0 and z and $A(x)$ is the cumulative distribution function for the inter-arrival time of the packets.

3.3 The queue length probabilities

We use these transition probabilities in order to compute the equilibrium probabilities for the queue length, as seen by an arriving packet. How to study the stability and the positive recurrency of the chain is described in [2], so that it can be verified that these probabilities are well defined. The transition probability matrix for the embedded Markov chain has the form

$$P = \begin{pmatrix}
\sum_{i=2}^{\infty} P_i & P_0 + P_1 & 0 & 0 & 0 & \ldots \\
\sum_{i=2}^{\infty} P_i & P_1 & P_0 & 0 & 0 & \ldots \\
\sum_{i=3}^{\infty} P_i & P_2 & P_1 & P_0 & 0 & \ldots \\
\sum_{i=4}^{\infty} P_i & P_3 & P_2 & P_1 & P_0 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots &
\end{pmatrix}$$

We denote by r the infinite dimensional vector whose components r_i's are the equilibrium probabilities for the embedded Markov chain describing the system, as it is seen by an arriving packet. This vector is a normalized left invariant eigenvector of the matrix P above and satisfies the infinite set of equations

$$\begin{cases} r = rP \\ \sum_{i=0}^{i=\infty} r_i = 1 \end{cases}$$

We can express the r_i's as follows :

$$\begin{aligned}
r_0 &= \frac{P_0\,(1-\xi)\,\xi}{\xi\,P_0 + \xi^2\,P_1} \\
r_i &= \frac{(P_0+P_1)\,(1-\xi)\,\xi^{i+1}}{\xi\,P_0 + \xi^2\,P_1} \quad for\, i \geq 1
\end{aligned}$$

where ξ is the unique solution in the interval $(0,1)$ of the equation in x

$$x = \sum_{i=1}^{\infty} (1 - \sum_{k=0}^{i-1} P_k)\, x^i$$

Conditions for the existence and uniqueness of this solution were given in [2]. The equilibrium probabilities p_i's for the queue length as seen by an arriving packet are finally

$$\begin{aligned}
p_0 &= r_0 + r_1 \\
p_i &= r_{i+1} \quad for\, i \geq 1
\end{aligned}$$

4 The voice arrival process

Since our method is based on the assumption that the arrival process is a renewal process, we present here a method of approximation of the voice arrival traffic by a renewal process. The cumulative distribution function of a single inter-arrival time in the voice superposition process was shown [2] to be given by

$$1 - F_N(t) = \begin{cases} (1-\lambda t)^{N-1} & t \leq T \\ \lambda^{N-1} (1-p)^N \psi(t-T)^{N-1} \phi(t-T) & t \geq T \end{cases}$$

where $\psi(t)$ and $\phi(t)$ are defined by

$$\begin{aligned} \psi(t) &= \frac{p_1}{\alpha_1} e^{-\alpha_1 (t-T)} + \frac{p_2}{\alpha_2} e^{-\alpha_2 (t-T)} \\ \phi(t) &= p_1 e^{-\alpha_1 (t-T)} + p_2 e^{-\alpha_2 (t-T)} \end{aligned}$$

with

$$\begin{aligned} p_1 &= \frac{\beta_1}{\beta_1 + \beta_2} \\ p_2 &= \frac{\beta_2}{\beta_1 + \beta_2} \end{aligned}$$

and

$$p = 1 - T(\beta 1 + \beta 2)$$

To model this process as a renewal process, we use the stationary-interval method introduced in [14]. We compute two or more moments μ_i's of F_N. The first one is easily given by

$$\mu_1 = \frac{1}{N\lambda}$$

where

$$\lambda^{-1} = T + (1-p)\left(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2}\right)$$

The second moment can be computed as follows :

$$\mu_2 = 2 \int_0^\infty t\,(1 - F_N(t))\,dt$$

which is

$$\begin{aligned} \mu_2 &= \frac{2\left[1 - (1-\lambda T)^N (1 + N\lambda T)\right]}{N(N+1)\lambda^2} \\ &\quad + \sum_{i=0}^{N} C_N^i \left(\frac{p_1}{\alpha_1}\right)^i \left(\frac{p_2}{\alpha_2}\right)^{N-i} \frac{2\lambda^{N-1}(1-p)^N}{N(i\,\alpha_1 + (N-i)\,\alpha_2)} \end{aligned}$$

where the C_N^i are the binomial coefficients. The squared coefficient of variation c^2 is finally given by

$$c^2 = \frac{\mu_2 - \mu_1^2}{\mu_1^2}$$

When c is larger than unity, the best approximation is obtained from a mixture of exponentials [14]. The approximate density function chosen for the description of the arrival process is then of the form

$$A(t) = q_1 \gamma_1 e^{-\gamma_1 t} + q_2 \gamma_2 e^{-\gamma_2 t}$$

where the parameters involved in the description are given by

$$\begin{aligned} q_i &= (1 \pm \sqrt{(c^2 - 1)/(c^2 + 1)})/2 \\ \gamma_i &= 2\, q_i\, \mu_1^{-1} \end{aligned}$$

When c is smaller than unity, we can use the sum of two independent exponential variables. We obtain the following approximate density function for the description of the arrival process

$$A(t) = q_1\, \gamma_1\, e^{-\gamma_1 t} + q_2\, \gamma_2\, e^{-\gamma_2 t}$$

where the parameters involved in the description are given by

$$\begin{aligned} \gamma_i^{-1} &= \frac{\mu_1}{2}(1 \pm \sqrt{2\,c^2 - 1}) \\ q_1 &= \frac{\gamma_2}{\gamma_2 - \gamma_1} \\ q_2 &= \frac{\gamma_1}{\gamma_1 - \gamma_2} \end{aligned}$$

A numerical computation showed that both these two cases had to be considered. Indeed, when the number of voice sources increases, c steps from values above unity to values below unity.

5 The data arrival process

We start by using a common model for the data arrival traffic : we assume that the data packets are generated according to a Poisson process, with rate ν. The inter-arrival time distribution corresponding to such a process is exponential with the same parameter ν. The squared coefficient of variation is equal to unity and the corresponding approximating renewal process is the process itself. We can therefore rewrite the inter-arrival time probability density function for the approximating renewal process as

$$\phi(t) = g_1\, \nu_1\, e^{-\nu_1 t} + g_2\, \nu_2\, e^{-\nu_2 t}$$

where the following parameters

$$\begin{aligned} g_i &= 1/2 \\ \nu_i &= \nu \end{aligned}$$

play the same role as the parameters q_1, q_2, γ_1 and γ_2 in the preceding section.

6 The approximated aggregate arrival process

We assume that the voice and data arrival processes are independent. We now have to approximate the superposition of two independent renewal processes. As pointed in [14], this is an approximation because the successive intervals in a renewal process are independent while the successive intervals in the superposition process are not. Indeed, the superposition of two renewal processes is itself a renewal process if and only if all three processes are Poisson [3], which is not the case here.

6.1 Stationary-Interval method

We first use the stationary-interval method to compute the first moment m_1 and the second moment m_2 of the distribution of the inter-arrival time for the approximating renewal process.

We denote by μ_{11} and μ_{21} the respective means for the inter-arrival times for the voice and data processes. We have

$$\mu_{11} = [T + (1-p)(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2})]/N$$

and

$$\mu_{21} = \nu^{-1}$$

This gives the first moment m_1 as

$$m_1 = \frac{T + (1-p)(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2})}{N + \nu(T + (1-p)(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2}))}$$

which is the inverse of the total arrival rate. To compute the second moment, we use the results presented in [14] and obtain finally

$$m_2 = \frac{2}{\mu_{11} + \mu_{21}} \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{q_i\, g_j}{\gamma_i\, \nu_j\, (\gamma_i + \nu_j)}$$

6.2 Asymptotic method

We can also choose to take into account the dependence between the successive intervals in the superposition process and use the asymptotic method for the computation of these two moments. Recall [14] that we have

$$\begin{aligned} \gamma_{11} &= \mu_1^{-1} \\ \gamma_{12} &= \mu_1^{-3}\,(\mu_2 - \mu_1^2) \end{aligned}$$

for the first and second cumulants of the voice process. We obtain

$$\begin{aligned} \gamma_{11} &= \frac{N}{T + (1-p)(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2})} \\ \gamma_{12} &= [\frac{2\,[1 - (1-\lambda T)^N\,(1 + N\,\lambda\,T)]}{N\,(N+1)\,\lambda^2} \\ &\quad + \sum_{i=0}^{N} C_N^i\,(\frac{p_1}{\alpha_1})^i\,(\frac{p_2}{\alpha_2})^{N-i}\,\frac{2\,\lambda^{N-1}\,(1-p)^N}{N\,(i\,\alpha_1 + (N-i)\,\alpha_2)}] \\ &\quad (\frac{N}{T + (1-p)(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2})})^3 - \frac{N}{T + (1-p)(\frac{p_1}{\alpha_1} + \frac{p_2}{\alpha_2})} \end{aligned}$$

We now need only compute the corresponding time-average limit cumulants γ_{2i} for the data process. Using Smith's formulaes [11], we obtain the following results

$$\begin{aligned} \gamma_{21} &= \nu \\ \gamma_{22} &= \nu \end{aligned}$$

We add the time-average limit cumulants for the voice and data process and obtain the time-average limit cumulants γ_1 and γ_2 for the superposition process as

$$\begin{aligned} \gamma_1 &= \gamma_{11} + \gamma_{21} \\ \gamma_2 &= \gamma_{12} + \gamma_{22} \end{aligned}$$

The moments m_1 and m_2 of the inter-arrival time distribution for the approximating renewal process are finally given by Smith's formulaes [14] as follows

$$\begin{aligned} m_1 &= \gamma_1^{-1} \\ m_2 &= m_1^3\,\gamma_2 + m_1^2 \end{aligned}$$

so that we obtain

$$m_1 = \frac{1}{\nu + \frac{N}{T+(1-p)(\frac{p_1}{\alpha_1}+\frac{p_2}{\alpha_2})}}$$

for the first moment, as well as

$$\begin{aligned} m_2 \;=\; & (\frac{1}{\nu + \frac{N}{T+(1-p)(\frac{p_1}{\alpha_1}+\frac{p_2}{\alpha_2})}})^3 \, [2\,\nu + \frac{1}{\nu + \frac{N}{T+(1-p)(\frac{p_1}{\alpha_1}+\frac{p_2}{\alpha_2})}} \\ & + \frac{2\,[1-(1-\lambda\,T)^N\,(1+N\,\lambda\,T)]}{N\,(N+1)\,\lambda^2} \\ & + \sum_{i=0}^{N} C_N^i \,(\frac{p_1}{\alpha_1})^i\,(\frac{p_2}{\alpha_2})^{N-i}\,\frac{2\,\lambda^{N-1}\,(1-p)^N}{N\,(i\,\alpha_1+(N-i)\,\alpha_2)} \\ & (\frac{N}{T+(1-p)(\frac{p_1}{\alpha_1}+\frac{p_2}{\alpha_2})})^3 - \frac{N}{T+(1-p)(\frac{p_1}{\alpha_1}+\frac{p_2}{\alpha_2})}] \end{aligned}$$

for the second.

6.3 Computation of the queue length distribution

We can now approximate the superposition arrival process as a renewal process, by using the first two moments and the method of approximation presented for the voice traffic in Section 4. We then use the method presented in Section 3 to compute the probability density function for the queue length in the multiplexer.

7 A more general model for the data traffic

Since the arrival traffic is complex and difficult to represent by common arrival models such as the Poisson model, we propose here the use of a more general model, the Packet Train model [9]. We apply our method of computation to a data traffic similar to that observed in [9].

The model can be described as follows [9]. We imagine that every node on the network is connected to every other node via a railroad track. Consider the track between two nodes A and B. A train consists of packets flowing on this track with the inter-packet (inter-car) time between them being smaller than a specified maximum, refered to as the maximum allowed inter-car gap (MAIG). If no packets are seen on the track for MAIG time units, the previous train is declared to have ended and the next packet is declared to be the locomotive (first car) of the next train. The inter-train time is defined as the time between the last packet of a train and the locomotive of the next train. We propose here an analytical study of our multiplexer, using the Packet Train model. The maximum allowed inter-car gap (MAIG) is denoted by M. We also denote by $\phi_1(t)$ the inter-train interval distribution as well as by $\phi_2(t)$ the inter-car interval distribution.

Our main assumption is that the number of cars in each train is geometrically distributed on the positive integers. This should characterize best the types of traffic for which the length of the trains is exponentially distributed and it seems to be coherent with the numerical results obtained in [9], showing that the autocorrelation function for the inter-arrival intervals between cars is very small so that successive time intervals can be considered as independent. Because of the lack of memory property associated with the geometric distribution, the packet stream on the line can be modeled as a renewal process. If we assume that there is a packetization time of T seconds, then each packet is followed by an interval of length $T+X$ where X is distributed as $\phi_1(t)$ with probability $1-p$ and as $\phi_2(t)$ with probability p. If we denote by θ_1 and θ_2 the first moments of $\phi_1(t)$ and $\phi_2(t)$, as well as σ_1 and σ_2 the respective variances, then the first moment of the inter-arrival time is given as

$$\mu_{21} = T + p\,\theta_2 + (1-p)\,\theta_1$$

while the second moment is

$$\mu_{22} = p(\sigma_2 + \theta_2^2) + (1-p)(\sigma_1 + \theta_1^2)$$

This enables us to compute the squared coefficient of variation for the inter-arrival time distribution, as

$$c^2 = \frac{p\,\sigma_2 + (1-p)\,\sigma_1}{(T + p\,\theta_2 + (1-p)\,\theta_1)^2}$$

so that if the coefficients of variation for the two distributions $\phi_1(t)$ and $\phi_2(t)$ are not ordered in the same way with respect to unity, by choosing different values for M, we can increase or decrease the value of p and change the value of c.

Using these moments and the value of c, we can apply the same approach as for the Poisson process, whether we choose the stationary-interval method or the asymptotic, and compute the probability density function for the queue length for the way it is seen by an arriving packet.

8 Final remarks

8.1 Another approach

Another approach for the approximation of the mixed traffic by a renewal process consists of proceding as for the case when the arrival traffic was composed of voice packets only. We follow the procedure presented in [14]. We define by $F_1(t)$ the cumulative distribution function for the inter-arrival periods on each voice line as well as $F_2(t)$ its equivalent for the data line. We assume that we have N voice lines feeding the multiplexer and that the data line represents the aggregate traffic for the data. We obtain the cumulative distribution function $F(t)$ for the inter-arrival times in the superposition process as follows

$$1 - F(x) = \sum_{i=1}^{N+1} (\lambda_i/\lambda)\,[1 - F_i(x)] \prod_{j \neq i} \lambda_j \int_x^\infty [1 - F_j(s)]\,ds$$

where the λ_i's represent the arrival rates on each line and λ is their sum. In our case, denoting by λ_1 and λ_2 the respective rates of arrival for the voice and data processes, we obtain

$$\begin{aligned} 1 - F(x) \;&=\; N\,(\lambda_1/\lambda)\,[1 - F_1(x)]\,\lambda_1^{N-1}\lambda_2\,\Big(\int_x^\infty [1 - F_1(s)]\,ds\Big)^{N-1} \int_x^\infty [1 - F_2(s)]\,ds \\ &\quad + (\lambda_2/\lambda)\,[1 - F_2(x)]\,\Big(\int_x^\infty \lambda_1\,[1 - F_1(s)]\,ds\Big)^N \end{aligned}$$

The first moment for the inter-arrival time is given as

$$m_1 = \lambda^{-1}$$

while the second moment is obtained as

$$m_2 = 2\int_0^\infty x\,(1 - F(x))\,dx$$

so that we can finally apply the same methods of approximation and approximate the aggregate arrival process by a renewal process.

8.2 Superposition of data traffic streams

In the sections above, we studied cases when the traffic is composed of one data line and a fixed number of voice lines. However, the procedure to adopt when we consider more than one line for the data traffic is slightly different. Indeed, when using the stationary-interval method, it is important to combine two processes at a time [14]. Other problems might also arise when using the asymptotic method. Indeed, in that second case, the number of processes involved is important and the method of computation has to be modified when this number increases [14]. Some work on the subject is presented in [3].

9 Summary

In this paper, we present methods of computation for the probability density function for the queue length in a Statistical Multiplexer fed by a mixed voice/data traffic. The equilibrium probabilities that we consider are not those that account for the average amount of time that the queueing process spends in each possible state, but rather those corresponding to the length of the queue as it appears to a packet at the instant of arrival.

The multiplexer is assumed to be fed by a data line along with a finite number of voice lines. The model used for the speech dynamics on each voice line is a three-state Markov chain model, yielding exponentially distributed active periods and hyper-exponentially distributed silence periods. We consider two models for the data traffic, a classical Poisson model and a Packet Train model.

Analytical results are given for both these cases and finally, some comments are given for the analysis of the cases when several lines are feeding the multiplexer with different types of data traffic.

10 Acknowledgements

This work was supported by the U.S. Army Research Office under Grant DAAL 03-89-K-0074 and by the National Science Foundation under Grant ECS-87-13598.

References

[1] D. Anick, D. Mitra and M. Sondhi, "Stochastic theory of a data-handling system with multiple sources" in *The Bell System Technical Journal*, Vol. 61, No. 8, October 1982.

[2] Frank M. Brochin and John B. Thomas, "Statistical Multiplexing in broadband ISDN : a study of delay performances" in *Proceedings of the Singapore International Conference on Networks 1989*, Singapore, July 1989.

[3] Ehran Cinlar "Superposition of point processes" in *Stochastic point processes : statistical analysis, theory and applications*, P. Lewis, ed., J.Wiley and Sons, 1972.

[4] John N. Daigle and Joseph D. Langford, "Queueing analysis of a packet voice communication system" in *Proceedings of the IEEE Infocom'85*, Washington, DC, March 1985.

[5] B. Eklundh *et al* "Asynchronous Transfer Modes : options and characteristics" in *Proc. of the 12-th ITC*, Torino, June 1988.

[6] Y. C. Jenq, "Approximations for packetized voice traffic in statistical multiplexers" in *Proceedings of the IEEE Infocom'84*, April 1984.

[7] Paul Kirton, John Ellershaw and Morgan Littlewood, "Fast packet switching for integrated network evolution" in *Proceedings of the International Switching Symposium*, March 1987, Phoenix, pp. B6-2-1.

[8] Leonard Kleinrock *"Queueing Systems : Volume 1, theory"*, John Wiley and Sons, 1975.

[9] Raj Rain and Shawn Routhier, "Packet trains : measurements and a new model for computer network traffic", in *IEEE Journal on Selected Areas in Communications*, No. 6, September 1986, pp. 986–995.

[10] Hiroshi Saito and Konosuke Kawashima "A survey on the analysis of integrated voice and data communication networks" in *Stochastic Analysis of Computer and Communication Systems*, North Holland, to appear.

[11] W.L. Smith, ''On the cumulants of renewal processes' in *Biometrika*, Vol. 46, pp. 1–29.

[12] Thomas E. Stern, ''A queueing analysis of packet voice'' in *Proceedings of the IEEE Global Telecommunications Conference*, San Diego, CA, December 1983.

[13] Thomas Stern, Nanying Yin and San-Qi Li ''Delay-fidelity tradeoffs in packet voice systems'' in *Proc. of the 12-th ITC*, Torino, June 1988.

[14] Ward Whitt ''Approximating a Point process by a renewal process : two basic methods'' in *Operations Research*, Vol. 30, Jan.-Feb. 1982.

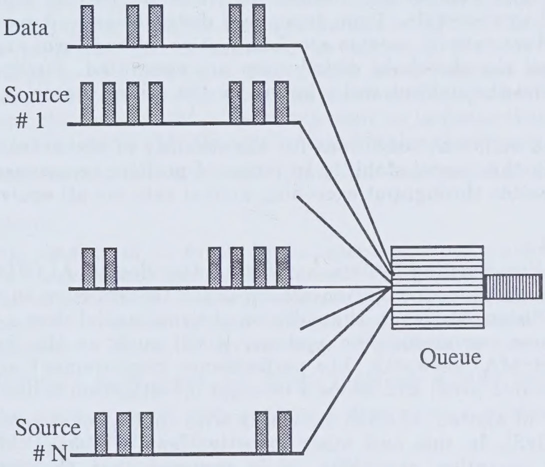

Data
Source # 1
Source # N
Queue

Figure 1

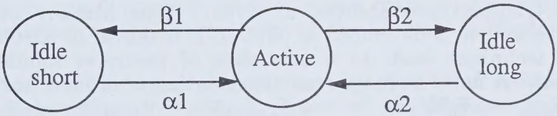

Idle short
Active
Idle long
β1
β2
α1
α2

Figure 2

TRANSIENT AND EQUILIBRIUM ANALYSIS OF SPREAD SPECTRUM SLOTTED ALOHA NETWORKS: FINITE MEMORY RECURSIONS

GARIMELLA RAMA MURTHY and EDWARD J. COYLE
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Abstract

A continuous time Markov chain model of slotted ALOHA networks with multipacket reception capability is developed. The packet lengths are allowed to assume a probability distribution of phase type so that an arbitrary distribution, including the constant packet length case, can be approximated.

A new class of recursive solutions called *finite memory recursions* are developed for the *transient and equilibrium* analysis of *finite as well as infinite* population networks modeled by the chain. Time dependent distribution and expected number of busy users, departure rate of packets are found. Also, closed form expressions for the expected delay and the threshold distribution are computed. Furthermore an exact upper bound for the throughput and a bound on the tail of the delay distribution are found.

Necessary and sufficient conditions for the stability of the network are obtained. It is shown that in this model stability in terms of positive recurrence, expected drift and maximum possible throughput exceeding arrival rate are all equivalent.

1. Introduction:

In a packet broadcasting system employing the slotted ALOHA protocol, it is assumed that if more than one station attempts for transmission in the current slot, then there is a collision. However, the collision channel model does not hold in many important multiuser communication systems [GvS] such as the Spread Spectrum Multiple Access (SSMA) networks. The performance improvement achieved by these systems is considerable [Rcd] and hence a detailed investigation is desirable.

The stability of slotted ALOHA networks with multipacket reception capability is discussed in [GvS]. In this and other investigations of slotted ALOHA networks with multipacket reception capability, it is assumed that the the slot length is constant. But in the model proposed in this paper the packet length is allowed to have a phase type distribution which can be chosen to approximate any distribution including a constant packet length arbitrarily closely.

The spread spectrum slotted ALOHA network is modeled by a G/M/1 type Markov process, whose state space structure allows the equilibrium probabilities to be computed recursively. One possible recursive solution is the matrix geometric solution developed in [Neu]. This solution has many nice numerical properties, but can usually not be found in closed form.

Motivated by a quest to find a closed form computable recursion for the equilibrium and transient probabilities of G/M/1 type Markov processes, a state space expansion technique is developed in [GrC1]. It is shown in [GrC2] that this state space expansion technique leads to a new class of recursive solutions called *finite memory recursions.* A finite memory recursive solution of order k has the form

$$\overline{\pi}_{n+k} = \overline{\pi}_n \mathbf{W}_1 + \cdots + \overline{\pi}_{n+k-1} \mathbf{W}_k \quad \forall \quad n \geq m-k.$$

It is also shown that a similar recursion holds for the Laplace transform of the transient state occupancy probabilities. In this paper, it is shown that the state space expansion can also be utilized to arrive at a finite memory recursion for the equilibrium and transient probabilities of a *finite* state space G/M/1 type Markov process. Considering the fact that in the past finite queues are rarely shown to admit closed form solutions, this is a significant result.

The existence of a finite memory recursion for the *finite as well as infinite* state space G/M/1 type Markov processes is then capitalized in the *transient and equilibrium* analysis of finite as well as infinite population slotted ALOHA networks. The time dependent and equilibrium performance measures obtained include throughput, distribution and moments of number of busy users, threshold distribution, exact closed form expression for the expected delay. Also, various performance measures of interest in the overload control of the slotted ALOHA networks are defined.

2. Modeling of Slotted ALOHA Networks with Multiple Successes in a Slot:

A1. Protocol Description:

In this protocol, unlike the slotted ALOHA protocol, simultaneous transmission of several packets within a slot does not necessarily result in the destruction of all the transmitted information. Specifically, we consider the case where any fraction of a fixed finite number (m-1) of packets are correctly received with positive probability and all packets are destroyed if the users exceed that finite number. The model can easily be considerably generalized to include the one in [GvS]. Also, the slot length is allowed to be a variable. This is achieved by allowing the slot length to have a probability distribution of phase type. The results for a constant slot length case can be achieved as a special case. In this case, the users can be synchronized by having a separate busy tone channel on which users transmit to indicate that they are sending a packet. Each user listens to this channel to synchronize their transmissions.

The following assumptions are made about the user population and the channel operation.

A2. User Population:

The network consists of N mutually independent and synchronized users who communicate over a common broadcast channel. Each user is allowed to hold only one packet in his buffer (Single buffer assumption). A user is considered to be busy from the moment it receives a new packet for transmission until after it is successfully transmitted. The inter arrival time of packets to each busy user has an exponential distribution with mean $\frac{1}{\lambda}$. When there are n busy users on the channel at the end of the current guard slot, each user transmits a packet in the following transmission slot with probability p.

A3. Channel Operation:

The time between the end of a slot and the beginning of the next slot is called as guard time. During this period the users synchronize their transmissions. This random variable has an exponential distribution with representation (1,-g). The length of a packet PS is a random variable with PH distribution with representation (α_P, P). The packets from each user will fill up the slot and this can be achieved by broadcasting the packet length along with the synchronization information.

B. Stochastic Model:

The channel activity is modeled using a stochastic process (n_t, cs_t, ph_t), where $n_t \triangleq$ number of busy users at time t, $cs_t \triangleq$ state of the channel at time t and $ph_t \triangleq$ phase index. There are 2 possible channel states, namely, the guard state and the successful transmission state. The activity on the channel is illustrated in Fig. 1 for the case where m=3.

It is easy to see that the process (n_t, cs_t, ph_t) is a continuous-time, time-homogeneous Markov chain. The state space of the Markov process is given by $(\,(i,j,k) : i \geq 0,\ 1 \leq j \leq 2,\ 1 \leq k \leq k(n) < \infty\,)$. The set of states with i=n is called as level n. The state transition diagram of the Markov process for the case where all the states are exponentially distributed is shown in shown in Fig. 2. From this diagram, it is apparent that the set of states in which n users are busy communicates only with the states in which n+1 users are busy and the set of states $\{\,(n+i,j): 1 \leq i \leq m-1,\ 1 \leq j \leq 2\,\}$. It is easy to see that this is the case even when the states have arbitrary phase type distributions.

The global balance equations corresponding to each state of the vector Markov process (n_t, cs_t, ph_t) can be found. Then the generator corresponding to the global balance equations can be shown to be

$$\mathbf{Q}=\begin{bmatrix} \mathbf{B}_0 & \mathbf{C}_0 & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{B}_1 & \mathbf{C}_1 & \mathbf{A}_0(1) & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{B}_2 & \mathbf{A}_2(2) & \mathbf{A}_1(2) & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{B}_3 & \mathbf{A}_3(3) & \mathbf{A}_2(3) & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{B}_{m-1} & \mathbf{A}_{m-1}(m-1) & \mathbf{A}_{m-2}(m-1) & \cdots & \mathbf{A}_0(m-1) & \cdots & \mathbf{0} \\ \mathbf{B}_m & \mathbf{A}_m(m) & \mathbf{A}_{m-1}(m) & \cdots & \mathbf{A}_1(m) & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{A}_1(N) \end{bmatrix}. \tag{2.1}$$

where

$$\mathbf{A}_0(n) = \begin{bmatrix} G(n)\mathbf{e}\alpha_p & \mathbf{0} \\ \mathbf{0} & G(n) \end{bmatrix}, \ \mathbf{A}_1(n) \ = \begin{bmatrix} \mathbf{P}-G(n)\mathbf{I} & p_0\mathbf{P}^{\circ} \\ \lambda_g\alpha_p & -(G(n)+\lambda_g) \end{bmatrix} \tag{2.2}$$

and for $n \geq m+1$, $2 \leq k \leq m$

$$\mathbf{A}_k(n)=\begin{bmatrix} \mathbf{0} & p_{k-1}\mathbf{P}^{\circ} \\ \mathbf{0} & 0 \end{bmatrix}, \ \mathbf{B}_0=-N\lambda, \ \mathbf{C}_0=[0 \ N\lambda], \ \mathbf{B}_i=\begin{bmatrix} p_{i-1}\mathbf{P}^{\circ} \\ 0 \end{bmatrix} \text{ for } 1 \leq i \leq m. \tag{2.3}$$

$$\mathbf{C}_1 = \mathbf{A}_1(n) \text{ for any } n \geq m+1 \text{ and } \mathbf{A}_j(k) = \begin{bmatrix} \mathbf{0} & p_{k-1}P^{\circ} \\ \mathbf{0} & 0 \end{bmatrix} \text{ for } 2 \leq k,j \leq m. \tag{2.4}$$

A continuous-time vector Markov process with the generator of the form in (2.1) is called a finite state space Markov process of G/M/1 type. The transient and equilibrium analysis of these Markov processes is now considered.

3. Finite Memory Recursive Solutions for the Equilibrium and Transient Analysis of Finite State Space G/M/1 type Markov Processes: Equilibrium Analysis:

A state space expansion technique is now described which will lead to a recursive solution for the *transient and equilibrium* probabilities of a finite state space G/M/1 type Markov process in which there is only one state at each level which receives a downward transition. This technique was developed in full detail in [GrC1], for the transient and equilibrium analysis of G/M/1 type Markov processes. It is motivated by a related technique in [BeC] for quasi-birth-and-death processes.

The technique is based on the notion of a *Level Entrance Direction Information Complete State Space,* in which there are no states on any level which receive transitions from both the levels above the level below. This technique is illustrated in Fig. 3.

After the expansion of the state space has taken place and the column operations discussed in [GrC2] are performed on the new generator, a matrix $\tilde{\mathbf{Q}}$ is obtained. On utilizing the structure of $\bar{\pi}\tilde{\mathbf{Q}} = \mathbf{0}$ leads to a system of equations

$$[\overline{\pi}_{\mathrm{n}}\ \overline{\pi}_{\mathrm{n+1}}\ \cdots\ \overline{\pi}_{\mathrm{n+m}}]\begin{bmatrix} \mathbf{A}_{1\mathrm{R}}(\mathrm{n+1}) & \mathbf{A}_{0\mathrm{L}}(\mathrm{n+1}) \\ \mathbf{A}_{2\mathrm{R}}(\mathrm{n+2}) & \mathbf{A}_{1\mathrm{L}}(\mathrm{n+2}) \\ \vdots & \vdots \\ \mathbf{A}_{\mathrm{m-1R}}(\mathrm{n+m-1}) & \mathbf{A}_{0\mathrm{L}}(\mathrm{n+m-1}) \\ \mathbf{A}_{\mathrm{mR}}(\mathrm{n+m}) & \mathbf{A}_{1\mathrm{L}}(\mathrm{n+m}) \end{bmatrix} = \mathbf{0} \text{ for } 1 \le \mathrm{n} \le \mathrm{N-m}. \quad (3.1)$$

and a boundary system of linear equations. When $[\ \mathbf{A}_{\mathrm{mR}}(\mathrm{n+m}) : \mathbf{A}_{1\mathrm{L}}(\mathrm{n+m})\ ]$ is invertible, (3.1) is equivalent to

$$\overline{\pi}_{\mathrm{n+m}} = \overline{\pi}_{\mathrm{n}}\mathbf{W}_1(\mathrm{n+1}) + \overline{\pi}_{\mathrm{n+1}}\mathbf{W}_2(\mathrm{n+2}) + \cdots + \overline{\pi}_{\mathrm{n+m-1}}\mathbf{W}_{\mathrm{m}}(\mathrm{n+m-1}), \quad (3.2)$$

Thus the notion of LEDI completeness and the associated state space expansion technique provide a new class of recursive solutions for the equilibrium probabilities of the finite state space G/M/1 type Markov processes considered in the paper. They are described below.

Definition: *Finite memory recursive solution of order* k for the equilibrium probabilities has the following structure

$$\overline{\pi}_{\mathrm{n+k}} = \overline{\pi}_{\mathrm{n}}\mathbf{W}_1(\mathrm{n+1}) + \overline{\pi}_{\mathrm{n+1}}\mathbf{W}_2(\mathrm{n+2}) + \cdots + \overline{\pi}_{\mathrm{n+k-1}}\mathbf{W}_{\mathrm{k}}(\mathrm{n+k}), \quad (3.3)$$

where $\overline{\pi}_{\mathrm{n}}$ is the vector of limiting probabilities of the states on level n of the process and $\mathbf{W}_{\mathrm{i}}$, $1 \le \mathrm{i} \le \mathrm{k}$ are n×n matrices.

Even when $[\ \mathbf{A}_{\mathrm{mR}} : \mathbf{A}_{1\mathrm{L}}\ ]$ is not invertible, by using a procedure similar to the one in [GrC1], it can be shown that a finite memory recursive solution for the equilibrium probabilities exists. Now the extension of above results to the transient analysis of finite state space G/M/1 type Markov processes is briefly explained. It is based on the idea of converting a system of differential equations into algebraic equations by Laplace transformation. This approach was first used in [ZhC] for the transient analysis of quasi-birth-and-death processes.

Transient Analysis:

The transition function of a time-homogeneous continuous-time Markov chain satisfies the differential equation

$$\dot{\overline{\pi}}(\mathrm{t}) = \overline{\pi}(\mathrm{t})\ \mathbf{Q} \quad (3.4)$$

which on Laplace Transformation yields

$$\overline{\pi}(\mathrm{s})(\ \mathbf{Q} - \mathrm{s}\mathbf{I}\ ) = -\overline{\pi}(0). \quad (3.5)$$

Assume that the process at time **0** is in one of the states at level 0. Equivalently, $\overline{\pi}(0) = [\overline{\pi}_0(0) : 0 : 0 : \cdots]$.

By considering the linear system of equations in (3.4) and performing the same column operations as in the equilibrium analysis case, it can be shown that a finite memory recursive solution for the Laplace transform of state occupancy probabilities exists. In other words

$$\overline{\pi}_{\mathrm{n+m}}(\mathrm{s}) = \overline{\pi}_{\mathrm{n}}(\mathrm{s})\mathbf{W}_1(\mathrm{n+1})(\mathrm{s}) + \cdots + \overline{\pi}_{\mathrm{n+m-1}}(\mathrm{s})\mathbf{W}_{\mathrm{m}}(\mathrm{n+m})(\mathrm{s}). \quad (3.6)$$

Detailed development follows the same arguments as in [GrC1].

Equilibrium and transient analysis of G/M/1 type Markov processes by finite memory recursion essentially requires the computation of a stable associated matrix geometric recursion and the inversion of the Laplace transform. Since the associated matrix geometric recursion matrix **T** can have modes outside the unit circle, it can cause instabilities in the computation. Since these modes are unexcited by the initial probability vector, it is desirable to completely remove them from the recursion [GrC3].

4. Performance Analysis of Finite Population Spread Spectrum Slotted ALOHA Networks:

Now various performance measures related to the finite population slotted ALOHA networks are computed utilizing the finite memory recursions.

Equilibrium Analysis:

(a) Throughput Calculation: The throughput of the network is given by

$$T = \sum_{n=1}^{N} p(n,s)[p_1 + 2p_2 + \cdots + (m-1)p_{m-1}] \tag{4.1}$$

where p_i is the probability that there are i users successfully transmitting in the transmission slot. When there are n busy users on the channel, p_i's are given by

$$p_i = \sum_{j=i}^{m-1} [p(i\,|\,j)p(j)] \quad \text{where } p_j = \sum_{n=j}^{N} \binom{n}{j} p^j(1-p)^{n-j}. \tag{4.2}$$

(b) Number of Busy Users: The probability that there are n busy users in the network is given by

$$\bar{\pi}_n \mathbf{e} = \bar{\pi}_{n-k} \mathbf{W}_1(n-k+1)\mathbf{e} + \bar{\pi}_{n-k+1} \mathbf{W}_2(n-k+2)\mathbf{e} + \cdots + \bar{\pi}_{n-1} \mathbf{W}_k(n)\mathbf{e}. \tag{4.3}$$

A closed form expression for the expected number of busy users is given by

$$E(N) = \bar{\pi}_1 \mathbf{e} + \cdots + (m-k)\bar{\pi}_{m-k}\mathbf{e} + [\bar{\pi}_{m-k+1} \cdots \bar{\pi}_m][\mathbf{I} - \tilde{\mathbf{T}}^{N+k-m-1}][\mathbf{I} - \tilde{\mathbf{T}}]^{-1}\hat{\mathbf{e}} \tag{4.4}$$

where $\hat{\mathbf{e}}$ is a column vector whose first n components are ones and the remaining are zeros.

(c) Expected Delay: Given the expressions for the throughput and the expected number of busy users in the network, the expected delay can be calculated using the Little's Theorem. The formula is given by

$$E(D) = \frac{E(N)}{T} \tag{4.5}$$

An upper bound on the delay can be calculated using the Chebyshev inequality. It is given by

$$P[\, D \geq d_0 \,] \leq \frac{E(D)}{d_0} \tag{4.6}$$

(d) Threshold Distribution:

$$T_{n_0}(t) = \sum_{n \geq n_0} \bar{\pi}_n(t)\mathbf{e} = [\bar{\pi}_{n_0}\ \bar{\pi}_{n_{0+1}} \cdots \bar{\pi}_{n_{0+(l-1)}}][\mathbf{I} - \tilde{\mathbf{T}}^{N-n_0}][\mathbf{I} - \tilde{\mathbf{T}}]^{-1}\hat{\mathbf{e}} \tag{4.7}$$

Detailed development of all these and related results can be found in [GrC3].

(e) Stability Analysis:

Drift Analysis and Input-Output Packet Flow Balance Principle:

From the generator **Q** in (3.1) of the G/M/1 type Markov process associated with the slotted ALOHA network, it is clear that the diagonal entries of the generator are bounded. Thus the continuous time chain is regular. It can be uniformized to arrive at a discrete time Markov chain $\tilde{Y}_n = (\, Y_n^1, Y_n^2, Y_n^3 \,)$. Let b>0 be the scalar constant bounding the diagonal entries of **Q** and used in the uniformization technique.

The stability of $(\, n_t, cs_t, ph_t \,)$ is now analyzed from the viewpoint of the notion of mean drift associated with the discrete time Markov chain $\tilde{Y}_n$. Since the underlying Markov chain is three dimensional, the mean drift is a three dimensional vector. It is given by

$$d_i(\mathbf{n}) = E[\, Y_i(k+1) - Y_i(k)\,|\, Y_i(k) = \mathbf{n} \,] \quad \text{for } i = 1,2,3. \tag{4.8}$$

In the above expression **n** indicates the level number in the state space. The first component, indicative of the dynamics of the busy user population is considered in more detail and it is shown in [GrC3] that the input-output packet flow balance principle and the concept of expected drift are equivalent, when attention is restricted to this component.

Transient Analysis:

Now various transient performance measures are computed utilizing the finite memory recursions.

(a) Number of busy users: The Laplace transform of the probability, as a function of time, that there are n busy users in the system is given by

$$\overline{\pi}_n(s)\mathbf{e} = \overline{\pi}_{n-k}(s)\mathbf{W}_1(n-k+1)(s)\mathbf{e} + \cdots + \overline{\pi}_{n-1}(s)\mathbf{W}_k(n)(s)\mathbf{e}. \tag{4.9}$$

The inverse transform of $\overline{\pi}_n(s)$ can be computed using some accurate inversion algorithm. Also, using (4.9), the Laplace transform of the expected number of busy users as a function of time, can be shown to be

$$E(N)(s) = \overline{\pi}_1(s)\mathbf{e} + \cdots + (m-k)\overline{\pi}_{m-k}\mathbf{e} + [\overline{\pi}_{m-k+1}(s)\ \overline{\pi}_2(s) \cdots \overline{\pi}_m(s)][\mathbf{I}-\mathbf{T}^{N-m}][\mathbf{I}-\mathbf{T}]^{-1}.$$

(b) Departure rate of packets from the network: The evolution in time of the rate at which packets depart the network (throughput) can be computed by inverting the following expression.

$$T(s) = \sum_{n=1}^{N} p(n,s)(s)[\, p_1(s) + 2p_2(s) + \cdots + (m-1)p_{m-1}(s)\,] \text{ where } p_i(s) = p\frac{(i)}{s} \tag{4.10}$$

Detailed development of these and some other related performance measures can be found in [GrC3].

5. Performance Analysis of Infinite Population SSSA Networks:

In this section, an infinite population network is considered. The case where the users transmit in each slot with probability p is considered. A finite load of new packets with an infinite population of users can be produced by letting the rate at which each individual user generates new packets to tend to zero in such a way that the following limit holds.

$$\lim_{\substack{N\to\infty \\ \lambda\to 0}} (N-n)\lambda = G \quad . \tag{5.1}$$

With this assumption, the matrices $\mathbf{A}_i$'s are independent of n and are given by

$$\mathbf{A}_0 = \begin{bmatrix} G\mathbf{e}\alpha_{\mathbf{p}} & \mathbf{0} \\ \mathbf{0} & G \end{bmatrix},\ \mathbf{A}_1 = \begin{bmatrix} \mathbf{P}-G\mathbf{I} & p_0\mathbf{P}^{\circ} \\ \lambda_g\alpha_{\mathbf{p}} & -(G+\lambda_g) \end{bmatrix} \text{ and } \mathbf{A}_k = \begin{bmatrix} \mathbf{0} & p_{k-1}P^{\circ} \\ \mathbf{0} & 0 \end{bmatrix} \text{ for } 2 \le k \le m. \tag{5.2}$$

Similarly, the boundary matrices can be computed from the ones in the finite population case. In these matrices, p_i is the probability that there are i successful transmissions in the current slot. They are computed using the expression in (4.2) by letting N tend to infinity.

The generator of the continuous time vector Markov process modeling the infinite population network is infinite dimensional and has the same form as in (2.1) with the matrices being independent of level number. Thus the slotted ALOHA network with an infinite population of users can be modeled as a G/M/1 type Markov process.

It is shown in [GrC1] that the state space of a G/M/1 type Markov process can be expanded so that a finite memory recursive solution exists for the equilibrium and transient state occupancy probabilities. In the following, for notational convenience, it is assumed that a finite memory recursive solution of order k results for the equilibrium and transient probabilities.

Various equilibrium and transient performance measures for the infinite population model of spread slotted ALOHA networks can be defined and computed in the same manner as in the finite population case.

Now various approaches to determine the stability of the infinite population networks are considered and an effort is made to unify the various approaches.

The slotted ALOHA network can be considered to be stable, when the number of busy users remains finite for all time. This condition holds true when the G/M/1 type Markov process modeling the network is recurrent nonnull. A necessary and sufficient condition for the chain to be recurrent nonnull is described in the following Theorem.

Theorem 5.1: The spectral radius of **R** is less than unity and the G/M/1 type Markov process (n_t, cs_t, ph_t) is recurrent nonnull if and only if

$$\overline{\pi}\mathbf{A}_0\mathbf{e} < \sum_{k=2}^{m} [(k-1)\overline{\pi}\mathbf{A}_k\mathbf{e}], \tag{5.3}$$

where $\overline{\pi}$ is uniquely determined by $\overline{\pi}\mathbf{A} = \mathbf{0}$ and $\overline{\pi}\mathbf{e} = 1$, with $\mathbf{A} = \mathbf{A}_0 + \cdots + \mathbf{A}_m$.

Proof. Refer [Neu].

An alternative characterization of recurrent nonnullness of G/M/1 type Markov processes through the finite memory recursions is provided in [GrC3].

The expression (5.3) is now interpreted by partitioning the vector $\overline{\pi}$ into subvectors corresponding to each channel state. We have $\overline{\pi} = [\ \overline{\pi}_s,\ \overline{\pi}_g\ ]$.

The left hand side of (5.3), on using the definition of $\mathbf{A}_0$ is equivalent to the rate at which packets arrive onto the channel, G. The right hand side of (5.3) is equal to $\overline{\pi}_s[\ p_1 + 2p_2 + \cdots + (m-1)p_{m-1}\ ]P^{\circ}$. An interpretation for this expression is given in the following Theorem.

Theorem 5.2: Consider the slotted ALOHA network modeled by the process **Q**. Fix all the other parameters of the network and increase the arrival rate of new packets, G until the inequality in (5.3) is converted into an equality. Each time the value of G is changed, the throughput T(G) from (4.1) is computed. Let the load which produces the equality be called $\hat{G}$. Then

$$\lim_{G \to \hat{G}} T(G) = [\ p_1 + 2p_2 + \cdots + (m-1)p_{m-1}\ ]\overline{\pi}_s P^{\circ} \frac{1}{\lambda_p}. \tag{5.4}$$

where $\frac{1}{\lambda_p}$ is the mean packet length. *Proof.*

The proof follows the standard ratio-limit theorem type argument and one can refer [CoL] for the details.

■

From Theorem 5.2 and (5.3), it can be inferred that stability is obtained when the maximum achievable throughput for the network is larger than the rate at which packets enter the channel. In the context of a discrete time Markov chain model, a similar result is obtained in [GvS] through the drift analysis techniques. It is shown in [GrC1] that in the model under consideration the stability condition in (5.3) and the condition resulting from drift analysis techniques are same and thus various stability notions pertaining to multiaccess networks are unified.

Throughput-Loading Curves: The right hand side of the inequality (5.3) can be computed using the probability,p_k that a packet involved in a collision of size j ($1 \leq j \leq m-1$) is successful. These probabilities are found in [MoL] for a direct sequence spread spectrum slotted ALOHA network. The right hand side can be shown [GrC3] to be

$$E[T]\overline{\pi}_s\mathbf{P}^{\circ} = \frac{1}{p}[\sum_{j=1}^{m-1} jp_j]\overline{\pi}_s\mathbf{P}^{\circ}, \tag{5.5}$$

where T is the random variable denoting the number of successful packets in the current slot given that more than one attempt for transmission. Now the throughput versus channel loading curves can be obtained by plotting $E[T]\overline{\pi}_s P^{\circ}\lambda_p$ against G. Any point below the curve is an achievable operating point for the protocol.

The analytical results relating to finite and infinite population spread spectrum slotted ALOHA networks are illustrated in the next section with specific example.

6. Sample Spread Spectrum Slotted ALOHA Network:

In this section, a spread spectrum slotted ALOHA network is considered in which the guard state random variable is exponentially distributed and the packet length is Erlang distributed with h stages. By letting the number of stages tend to infinity the performance of a network in which the slot length is constant can be obtained.

The Erlang packet length distribution has a representation (α_P, P) with mean $\frac{1}{\lambda_p}$. The guard state random variable has an exponential distribution with mean $\frac{1}{\lambda_g}$. The operating region of the protocol is calculated by evaluating the throughput expression in (5.5). Using the Theorem (5.2), the maximum possible throughput is given by

$$T_{\max} = \frac{1}{\lambda_p} \sum_{k=1}^{m-1} k p_k \overline{\pi}_s \mathbf{P}^{\circ} \tag{6.1}$$

$$= E[T] \frac{\lambda_g}{\lambda_p\left(\frac{\lambda_g}{G_{\max}}\left[\left(1+\frac{G_{\max}}{h\lambda_p}\right)^{h-1}-1\right]+1\right)}, \tag{6.2}$$

The expression in (6.2) allows the throughput of a SSSA network in which the slot length is constant to be evaluated. On letting h tend to infinity, the above expression becomes

$$T_{\text{constant slot length}} = E[T] \frac{\lambda_g}{\lambda_p\left(1+\frac{\lambda_g}{G_{\max}}\left(e^{\frac{G_{\max}}{\lambda_p}}-1\right)\right.}). \tag{6.3}$$

It is also of interest to find the maximum possible throughput of a multipacket success slotted ALOHA protocol in which there is no guard state. By letting λ_g tend to infinity and $\lambda_p = 1$, the above expression becomes

$$T_{\text{multipacket slotted ALOHA}} = E[T] \frac{G_{\max}}{(e^{G_{\max}}-1)}. \tag{6.4}$$

7. Performance Measures for Overload Control of Spread Spectrum Slotted ALOHA Networks:

The network modeled by a continuous time Markov chain could be either stable or unstable. When the network is stable, it is of interest to know how fast the transient probability distribution reaches the equilibrium and when unstable, how fast backlogs are created. Some of the performance measures reflecting these characteristics are now defined which will eventually lead to the design of overload controls.

A. Ergodic case:

The evolution in time of the the distribution of the number of busy users, expected number of busy users and the state occupancy probability of the states resemble the *impulse response* of a linear system. Thus, it is most natural to characterize the time dependent behavior of the network by the following measures.

(a) Settling Time: It is the time it takes for the system to stay within a certain percentage (typically 2%) of the steady-state response.

Settling time provides some guidance as to how often the adaptive control strategies for retransmission probabilities be updated. An approximation to the settling time of an infinite waiting space M/M/1 queue is found in [Ste].

(b) Raise Time: It is the time required for the quantity of interest to rise from 10% to 90% of the final value.

(c) Peak Overshoot: It is the normalized difference between the peak and the steady state value of the quantity of interest.

(d) Rate of change of the expected number of busy users: When the system starts in the zeroth level, the expected number of busy users normally increases monotonically to the steady state value. Thus the derivative of this quantity is a natural measure of how fast the system reaches the steady state value.

These measures can be computed using the transient probabilities determined through the finite memory recursions and their accuracy is limited by the numerical inversion algorithm.

B. Non-ergodic Case:

In this case, the transient distribution of number of busy users, expected number of busy users and the other moments become unbounded. Some of the measures characterizing how fast the system creates backlogs can be defined along the same lines and details can be found in [GrC3].

References:

[BeC] S. L. Beuerman and E. J. Coyle, "State space expansions and the limiting behavior of quasi-birth-and-death processes," *Advances in Applied Probability,* 21, June 1989.

[CoL] E. J. Coyle and Bede Liu, "A Matrix Representation of CSMA/CD Networks," *IEEE Transactions on Communications,* vol. 33, pp. 53-64, Jan 1985.

[Jen] Y. C. Jenq, "On the stability of Slotted ALOHA Systems," *IEEE Transactions on Communications,* vol. 28, Nov. 1980.

[MoL] R. K. Morrow and J. S. Lehnert, "Bit-to-Bit Error Dependence in Slotted DS/SSMA Packet Systems with Random Signature Sequences," Proceedings of the 22nd Conference on Information Sciences and Systems, March 1988.

[Neu] M. F. Neuts, "Matrix-Geometric solutions in stochastic models," Johns Hopkins University Press, Baltimore.

[GrC1] G. Rama Murthy and E. J. Coyle, "Finite Memory Recursive Solutions in Stochastic Models: Equilibrium and Transient analysis," Technical Report-29, Purdue University.

[GrC2] G. Rama Murthy and E. J. Coyle, "Finite Memory Recursive Solutions for the Transient and Equilibrium Analysis of G/M/1 type Markov Processes," submitted to the *IEEE Transactions on Information Theory.*

[GrC3] G. Rama Murthy and E. J. Coyle, "Transient and equilibrium analysis of slotted ALOHA networks with multipacket reception capability," Manuscript in Preparation.

[Rcd] D. Raychaudhuri, "Performance Analysis of Random Access Packet Switched Code Division Multiple Access Systems," *IEEE Transactions on Communications,* vol. COM-29, pp. 895-901, 1981.

[Ste] T. E. Stern, "Approximations of Queue Dynamics and Their Application to Adaptive Routing in Computer Communication Networks," *IEEE Transaction on Communications,* vol. 27, pp. 1331-1335, September 1979.

[ZhC] J. Zhang and E. J. Coyle, "Transient analysis of quasi-birth-and-death processes," *Communications in Statistics---Stochastic Models,* vol. 5, No. 3, August-Sept. 1989.

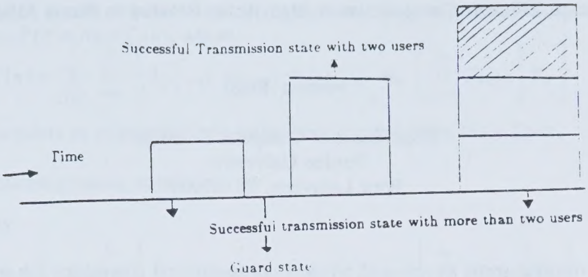

Successful Transmission state with two users
Time
Successful transmission state with more than two users
Guard state

Fig 1. Typical channel activity of a spread spectrum slotted ALOHA network.

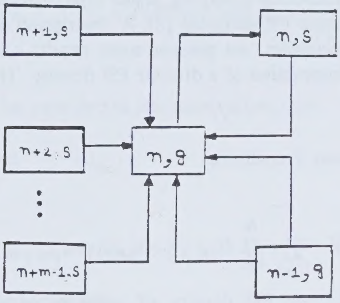
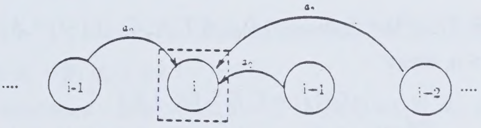
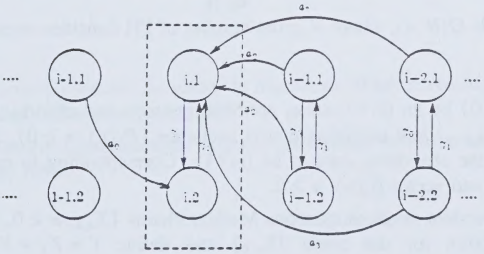

n+1, s
n, s
n+2, s
n, g
n+m-1, s
n-1, g

Fig 2. State transition diagram for the spread spectrum slotted ALOHA protocol. All possible transitions into and between all states in which n users are busy are shown.


a₀
a₁
a₂
....
i-1
i
i−1
i−2
....

LEDI Incomplete G/M/1-Type Markov Process


....
i-1.1
i.1
i−1.1
i−2.1
....
....
i-1.2
i.2
i−1.2
i−2.2
....

LEDI Compiete G/M/1-Type Markov Process

Fig 3. State space expansion of finite state space G/M/1 type Markov process.

Some Efficient Computational Algorithms Related to Phase Models

Vernon Rego

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract: We develop some simple and efficient computational algorithms for some models that utilize phase type distributions. In essence, we exploit the geometric nature of PH-densities to reduce certain exponential time computations to linear time, using recursive schemes.

1. Motivation

In a variety of applications involving phase type distributions [1], it is necessary to compute convolutions of discrete PH densities [2]. A description of PH distributions can be found in [1]. In the following discussion, we present some results on efficient computational schemes, using (αT) to be a respresentation of a discrete PH density. The main results are as follows:

Theorem 1 (Homogeneous Transitions)

The quantity

$$C(n) = \sum_{D(n)} \prod_{k=1}^{N} [I_{\{n_k = 0\}}\, \alpha_{k,m+1} + I_{\{n_k > 0\}}\, \alpha_k\, T_k^{n_k - 1}\, \beta_k] \tag{1}$$

where (α_k, T_k) is a discrete PH density of order m with initial vector $(\alpha_k, \alpha_{k,m+1})$, and $D(n) = \{(n_1, \ldots, n_N) \mid n_j \geq 0,\ 1 \leq j \leq N,\ \sum_{1}^{N} n_j = n\}$, is given by $\alpha_N\, C_N(n)\, \beta_1$, via the recurrence

$$C_i(j) = T_i\, C_i(j-1) + d_i\, A_i\, C_{i-1}(j) - d_i\, T_i\, A_i\, C_{i-1}(j-1) + A_i\, C_{i-1}(j-1) \tag{2}$$

for $2 \leq i \leq N$, $1 \leq j \leq n$, where

$$C_1(j) = T_1^{\,j-1} \quad \text{for } j \geq 1 \tag{3}$$

$$C_i(0) = d_i\, A_i\, \cdots\, d_1\, A_1 \quad \text{for } 1 \leq i \leq N\ , \tag{4}$$

$A_1 = I$, $A_i = \beta_i\, \alpha_{i-1}$ for $2 \leq i \leq N$, and $d_i = \dfrac{\alpha_{i,m+1}}{\alpha_i\, \beta_i}$ for $1 \leq i \leq N$. The run time complexity of this computation is $O(N. n)$, where N is the number of PH densities considered.

Let $\{X_{n,k}\,; n \geq 0\}$ be an $(m+1)$-state, nonhomogeneous and absorbing Markov chain with initial vector $(\alpha_k, \alpha_{k,m+1})$ and transition matrix sequence $\{P_k(n)\,; n \geq 0\}$, for each k, $1 \leq k \leq N$. As before, we take the absorbing state to be $(m+1)$. Corresponding to each matrix $P_k(n)$, we have a matrix $T_k(n)$ and vector $\beta_k(n)$, $n \geq 0$.

Given N independent nonhomogeneous Markov chains $\{X_{n,k};\ n \geq 0,\ 1 \leq k \leq N\}$, let Y_k be the time to absorption for the chain $\{X_{n,k}\}$ and define $Y = Y_1 + Y_2 + \cdots + Y_N$. The probability that all N chains require a total of n steps for their respective absorptions into state $(m+1)$ is given by $P(Y=n) = C(n)$, where

$$C(n) = \sum_{D(n)} \prod_{k=1}^{N} \left\{ I_{\{n_k=0\}}\, \alpha_{k,m+1} + I_{\{n_k>0\}}\, \boldsymbol{\alpha}_k \left[\prod_{j=0}^{n_k} T_k(j) \right] \boldsymbol{\beta}_k \right\} \tag{5}$$

which, like (1), requires an exponential algorithm if computed using brute force.

Theorem 2 (Nonhomogeneous transitions)

The quantity

$$C(n) = \sum_{D(n)} \prod_{k=1}^{N} \left\{ I_{\{n_k=0\}}\, \alpha_{k,m+1} + I_{\{n_k>0\}}\, \boldsymbol{\alpha}_k \left[\prod_{j=0}^{n_k} T_k(j) \right] \boldsymbol{\beta}_k \right\} \tag{6}$$

is given by $C_N(n)$, via the recurrence

$$C_N(n) = \sum_{k=0}^{n} \boldsymbol{\alpha}_N\, A_N(n-k)\, \boldsymbol{\beta}_N\, C_{N-1}(k) \tag{7}$$

with $C_0(0) = \boldsymbol{I}$, $C_0(k) = \boldsymbol{0}$, $A_k(0) = d_k \boldsymbol{I}$, $A_k(1) = T_k(0) = \boldsymbol{I}$, and $d_k = \dfrac{\alpha_{k,m+1}}{\boldsymbol{\alpha}_k\, \boldsymbol{\beta}_k}$, for $k \geq 1$.

The preceding discussion can also be extended to the multivariate case.

Theorem 3 (Multivariate Extension)

The quantity

$$C(a,b) = \sum_{F(a,b)} \prod_{k=1}^{N} \{ I_{\{m_k+n_k=0\}}\, \alpha_{k,m+1} + I_{\{m_k+n_k>0\}}\, \boldsymbol{\alpha}_k\, \boldsymbol{X}^{m_k}\, \boldsymbol{Y}^{n_k}\, \boldsymbol{\beta}_k \} \tag{8}$$

is given by $\boldsymbol{\alpha}_N\, C_N(a,b)\, \boldsymbol{\beta}_1$ via the recurrence

$$C_i(j,k) = \boldsymbol{X}_i[C_i(j-1,k) + (1-d_i)\, \boldsymbol{A}_i\, C_{i-1}(j-1,k)] + d_i\, \boldsymbol{A}_i\, C_{i-1}(j,k) \tag{9}$$

$$+\, \boldsymbol{Y}_i[C_i(j,k-1) + (1-d_i)\, \boldsymbol{A}_i\, C_{i-1}(j,k-1)] \tag{10}$$

for $1 \leq i \leq N$, $1 \leq j \leq a$, $1 \leq k \leq b$. The required quantities are given by $d_n = \dfrac{\alpha_{n,m+1}}{\boldsymbol{\alpha}_n\, \boldsymbol{\beta}_n}$ for $1 \leq n \leq N$, $\boldsymbol{A}_1 = I$, and $\boldsymbol{A}_n = \boldsymbol{\beta}_n\, \boldsymbol{\alpha}_{n-1}$ for $1 < n \leq N$.

Finally, boundary conditions give $C_0(0,0) = \boldsymbol{I}$, $C_i(0,0) = (\prod_{k=1}^{i} d_i)\, \boldsymbol{I}$, $C_i(0,k) = \boldsymbol{Y}_i^k\, \boldsymbol{A}_i$, $C_i(k,0) = \boldsymbol{X}_i^k\, \boldsymbol{A}_i$ for $1 \leq i \leq N$ and $k > 0$.

References

[1] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, The Johns Hopkins University Press, 1981.

[2] V. J. Rego, "Efficient Computational Algorithms Related to Phase Models," to appear in Acta Informatica, 1990.

A DECOMPOSITION ALGORITHM FOR THE STATE TRANSITION LATTICES OF CERTAIN NON-PRODUCT QUEUEING PROTOCOLS

CHENG-KUO HONG and IRVING Y. WANG
New Jersey Institute of Technology
Newark, New Jersey

ABSTRACT

In this paper, an algorithm is developed to decompose the state transition lattice of a class of non-product form queueing models. Here only the lattices of regular type A structure are considered. By applying the algorithm, the lattice is decomposed into solvable subsets which can be solved sequentially.

1. INTRODUCTION

In studying the behavior of computer systems, queueing network models are a useful tool for evaluating performance and studying the interaction of resources,software and workload [5]. The product form solution of the balance equations, when it exists, plays an important role in analyzing such queueing network models.

A great deal of research has been performed on the subject of product form solutions. Originally, Jackson described the equilibrium state probabilities for open networks of queues with a single class of jobs, Poisson arrival statistics, and exponential service times [4]. There probabilities were of a characteristic "product form" :

$$p(n_1, n_2, \cdots, n_m) = p_1(n_1)p_2(n_2)\cdots p_m(n_m) \tag{1}$$

Later, Gordon and Newell analyzed closed queueing networks with each station having an exponential service time distribution [3]. They obtained the equilibrium state probability as :

$$p(n_1, n_2, \cdots, n_m) = \frac{1}{G(N)} \prod_{i=1}^{m} f_i(n_i) \tag{2}$$

where *G(N)* is a normalization constant chosen to make all the feasible state probabilities sum to one. *N* is the total number of jobs. The f_i areakin to the marginal probabilities of (1). Later, the class of queueing networks with product form solution was further generalized in [1].

Product form solutions are of great interest because the direct numerical solution of balance equations is computationally expensive. While the calculation of *G(N)* may not be trivial [2], the evaluation of product form solutions is still computationally and analytically preferable to solving very large systems of linear equations.

Non-product form queueing networks are far less likely to have a closed form solution for the equilibrium probabilities. Direct solution techniques are prohibitively expensive. Techniques analogous to that of the z-transform can sometimes be used to determine distributions of interest [5]. In [8]. a class of non-product form queueing network is described for which an exact decomposition of the state transition lattice is possible. An algorithm is presenced in this paper for performing the decomposition in certain cases. This leads to a sequential method of solution with large computational savings. In some cases, recursive expressions for the equilibrium probabilities can be developed [7].

The sequential decomposition method for a class of non-product form queueing networks depends on the concept of a "solvable subset" of state:

Definition 1.1: a solvable subset of queueing network states is a subset of states for which the equilibrium probabilities can be determined without regard to the equilibrium probabilities of the remaining unknown states. Here probabilities are determined with respect to a reference probability.

Note also that the direct solution of linear equations takes time proportional to the cube of the number of equations. If *N* states can be solved as M subsets then the computational effect is proportional to $(N/M)^3 * M$ rather than N^3.

Two types of structure which allow the state transition lattice to be decomposed into solvable subsets will now be presented. The first type of structure is illustrated in Figure 1. Here each circular subset represents a state or a group of states. For the *ith* subset the rule is that there must be only one state, with unknown probability, external to the subset from which a transition(s) entering the subset originates. The subsets are solved sequentially, starting from the first subset to the second and so on. There is no restriction on the number of transitions which may leave the *ith* subset for destinations in the $j = i + 1, i + 2, \ldots$ subsets. This type of structure will be referred to as Type A structure.

The second type of structure is illustrated in Figure 2. Here the first subset consists of a single state. The remaining subsets each consist of a state or a group of states. There are arranged in a tree type of configuration with the flow between subsets from the top of the diagram to the bottom and a return flow from the bottom level back to the top level. The subsets may be solved from the top to the bottom. Transitions may traverse several levels (e.g. dotted line in the figure) as long as the direction of flow is downward. This type of structure will be referred to as Type B structure.

A great many protocols of practical interest have a lattice geometry analogous to that which has just been described.

The main purpose of this paper is to present an algorithm which can decompose a state transition lattice into a number of solvable subsets, each of which can be solved independently. The basic concept behind this algorithm is that all possible models of solvable subset are stored in the memory of computer,and referred to *prototypes.* Each prototype corresponds to a set of balance equations which can be used to determine the state probabilities for that prototype. Whenever a state transition lattice is to be examined, a group of states on this lattice is fetched and matched with the prototypes. This is only a match of the structure, not necessarily of size. The corresponding balance equations are obtained if a match occurs.

In this paper, only the lattices of a certain regular and rectangular structure are considered due to the limited size of computer memory.

Definition 1.2: a lattice of regular structure is defined as the lattice that is composed only by *regular solvable subsets.*

Definition 1.3: a regular solvable subset is defined as the solvable subset in which, except for the *bridging state* and the *boundary state*, every state contains only one special pattern of transitions that link this state with its neighboring states.

Definition 1.4: a bridging state is the state which bridges two solvable subsets.

Under the above definitions, the lattices considered in this paper always possess a structure similar to the one in Figure 3.

Fortunately, there are many computer network models with this structure.

Other constraints made in our model are that transitions in the lattice occur only between neighboring states and the diagonal transitions are not taken into consideration. This is due,again, to the reason that the memory size is limited.

2. PROTOTYPES

The introduction to the structure of prototypes in this section is based on the assumption that the reference state is located at lower left corner of the lattice, and the subsets are organized sequentially from the bottom row to the top row. In the matching process, each of these structures as well as its image reflection are rotated $0^o, 90^o, 180^o, 270^o$ clockwise during the comparison with the group of states to be examined. Details of the rotation will be discussed later.

A prototype consists of four parts:

1. Transitions between interior states. There are two possible types of transitions, shown in Figure 4.1.a,b, in this category. In Figures 4.1.a,b each cross represents a state and the leftmost one stands for the bridging state. These four states are connected in the fashion that all transitions are either bidirected or directed toward the bridging state. The states connected otherwise are excluded from the group because their probabilities are not involved in the balance equations corresponding to this structure.

2. Transitions between states in the currently considered prototype and states in the previously solved prototypes. Figures 4.2.a,b shows two possible types of such transitions.

vector		description
0	0	no transition
1	1	Inward transition originated from a state with unknown probability
1	-1	Inward transition originated from a state with known probability
0	1	Two-direction transition with its own probability unknown
0	-1	Two-direction transition with its own probability known
-1	1	Outgoing transition with its own probability unknown
-1	-1	Outgoing transition with its own probability known

Table 1: Definition of vetcor for state transition

Note that there is no transition directed from the currently considered prototype to the previously solved one because, otherwise, the latter can not be a solved subset because the unknown variables would have outnumbered the balance equations when it was considered.

3. Transition between states in the currently considered prototype and states in the unsolved prototypes. Again, two possible such transition are illustrated in Figures 4.3.a,b. Also note that there is no transition originating from the unsolved groups to the prototype under considerations.
4. Transitions at the bridging state. Four possible combinations are shown in Figures 4.4.a,b,c,d.

There are thirty two possible synthesized structures made up from the above four categories. However, eight of them contain transient states such as one of them in Figure 5, and they are disqualified as prototypes. Therefore, only twenty four prototypes are stored in the memory bank and shown in Appendix A of [10].

The reference state is possibly located at one of the other three corners (The algorithm for selecting the reference state is discussed in section 4). The bridging state, also, does not stand at the leftmost of a subset for all cases. Hence, the clockwise rotations of both prototypes and their image reflection are generated to create prototype clusters during the matching procedure is implemented. An example of this rotation is illustrated in Fig 6. The outcomes of the various rotations in one specified prototype correspond to balance equations of similar form except that the coefficients are varied. Appendix B of [10] list the balance equations for the example in Figure 6 and show how the coefficients are managed.

3. MATCHING PROCEDURE

Before advancing to the matching procedure, the state vector used to represent the incoming and/or the outgoing transitions of a certain state is introduced first.

The definition of state vector is shown in Table 1 Two binary digits are adopted to represent transition on one of the eight directions connected to a state. Thus, a state vector consists of four pairs of binary digits, each of which denotes the linkage with neighbor on the left, above, on the right, and below, respectively. For example, the reference state (RS),the state with unknown probability (US) and the state with known probability (KS) are shown in Figure 7. The reference state can be depicted by the vector as :

$$[0\ 1\ 1\ 1\ 1\ 1\ 0\ \ 1] \tag{1}$$

A subset vector is, then, a synthesis of state vectors in that subset. In the order the vector of bridging state is placed in the first position and the vector of boundary state on the other end in the fourth. Two more state vectors in the subset vector are sufficient to describe the subset and are placed in the second and third positions. All the prototypes are stored in the memory in the form of this 32-bit subset vector, as well as the square of vector magnitudes. The prototype in Figure 7 can be described by the vector :

[0 0 0 1 0 1 0-1 0-1-1 1 0 1 1-1 0 1-1 1 0 1 1-1 0 1-1 1 0 0 1-1]

and the square of this vector magnitude is :

$$1^2 + 1^2 + (-1)^2 + (-1)^2 + (-1)^2 + 1^2 + 1^2 + 1^2 + (-1)^2 + 1^2 + (-1)^2 + 1^2 + 1^2 + 1^2 + (-1)^2 + 1^2 + (-1)^2 + 1^2 + 1^2 + (-1)^2 = 20$$

Note that the number of states in a subset could be less than four, therefore, the magnitudes of first two and first three elements in the prototype vector are also parts of the stored.

After identifying the regular structure, the fetched subset is transformed into subset vector and ready for the matching. Let A(n) denote the nth prototype vector in the memory, and B denote the fetched subset vector. The matching procedure is performed as follows : those A(n)'s which have the same the square of magnitude as B are chosen to be further processed. Inner product $A(n) * B$ is compared against the square of magnitude of the nth prototype $|A(n)|$; If they are not equal, other members in this A(n) cluster are set to task, one by one. This procedure will be repeatedly applied on clusters of other chosen A(n)'s unless a match is found or n exceeds the number of stored prototypes.

4. ALGORITHM

In this section, the algorithm is discussed in a step-by-step fashion. A flow chart is also provided in Figure 8-9 to aid understanding. The algorithm consists of seven major steps.

Step 1: Find the reference state.

Due to the assumption that the lattice inspected is of regular structure, the reference state must be one of states at boundary corners. In this step, the corner state with only one incoming transition is selected as the reference state. If no such state can be located, the algorithm ceases going further, and outputs "no reference state". If more than one such states exist, the one first encountered by the algorithm is chosen. Probability of the reference state is temporarily set to 1.

Step 2: Determine if the currently considered subset is a single-state solvable subset.

Any subset which contains simply one state and meets the solvable subset requirement of Type A (ie. only one state, with unknown probability, exterior to this subset originates transitions into this subset) is referred to as a single-state solvable subset. One balance equation with one unknown is involved in the subset, this unknown can be determined easily. Thus, single-state solvable subsets are the most preferable during the whole process.

Step 3: Ascertain if the lattice is of Type A structure.

If the attempt of searching for single-state subset fails, more states will be drawn into the currently-considered subset to accompany the bridging state. In the lattice of regular structure, the bridging state always sits on the boundary. Figrue 10 depicts the basic structure of transitions linked with bridging state. One of the transitions originating from unknowns is followed as a path to drag in states. Every newly introduced state can bring in at most one unknown, which will be the next state introduced, in the subset. If the procedure fails in both paths, this lattice is declared not of Type A structure by the algorithm.

Step 4: Perform the matching procedure.

Details of this step has been described in the previous section.

Step 5: Solve the balance equations.

The balance equations of fetched subset, which are obtained through the matching procedure, are used to determine the probabilities (in term of probability of reference state) of states in a subset.

Step 6. Change the vector of the neighbors of the state(s) with known probability

5	57.21	52.91	39.76	26.85	16.87	9.97	5.50
4	74.39	44.02	25.31	14.12	7.65	4.04	2.08
3	98.66	58.09	32.66	17.60	9.16	4.65	2.33
2	136.42	78.67	42.05	21.24	10.32	4.92	2.36
1	207.88	109.65	52.03	23.30	10.11	4.36	1.94
0	399.42	143.66	51.69	18.63	6.75	2.50	1.00
	0	1	2	3	4	5	6

Table 2: The state probabilities before normalization

Once the probabilities of states are determined in the previous step, the vectors of neighbors of these states have to be changed according to Table 1. This will help the determination of succeeding matching procedures.

Step 7: Normalize the state probabilities

Sum of all the probabilities in a lattice must equal to one, therefore, normalization yields the exact values of the state probabilities.

5. EXAMPLE

A typical example of a type A lattice is shown in Figure 11, which is the state transition lattice of a priority queue. The rightmost state on the bottom row is selected as the reference state by our algorithm. The probabilities of all other states on the bottom row can be determined sequentially in a one-by-one fashion. Each row above the bottom one is a solvable subset with the leftmost state serving as the bridging state.

With $\lambda = 2$, $\lambda = 1$, $\mu = 4$, $\mu = 3$, the balance equations for the reference state, the boundary state on the bottom row, and other states on the bottom row can be written as

(0,6) : $(\mu + \lambda)P = \lambda P$

and

(0,0) : $(\lambda + \lambda)P = \mu P + \mu P$

(0,j) : $(\mu + \lambda + \lambda)Pj = \lambda Pj + \mu Pj$

$j = 1, \ldots, 5$

respectively.

Each row above the bottom one is matched with the first one of the twenty four prototypes in Fig 12, and corresponds to a set of balance equations

(1,0) : $(\mu + \lambda + \lambda)P = \lambda P + \mu P + \mu P$

(1,j) : $(\mu + \lambda + \lambda)Pj = \lambda Pj + \lambda Pj + \mu Pj$

(1,6) : $(\mu + \lambda)P = \lambda P + \lambda P$

$j = 1, \ldots, 5$

The top row, however, can be decomposed into single-state solvable subsets again, and the balance equations corresponding to those subsets are expressed as

(5,0) : $(\mu + \lambda)P = \lambda P + \mu P$

for bridging state, and

(5,j) : $(\mu + \lambda)Pj = \lambda Pj + \lambda Pj + \mu Pj$

(5,6) : $\mu P = \lambda P + \lambda P$

$j = 1, \ldots, 5$

for others.

The state probabilities can be determined by above equations, and the values are listed in Table 2 (before normalization) and Table 3 (after normalization).

Please refer to the Appendix C of [10] for the details of algorithm which are written in Fortran language.

6. CONCLUSION

5	0.0296	0.0274	0.0206	0.0139	0.0087	0.0052	0.0028
4	0.0385	0.0228	0.0131	0.0073	0.0040	0.0021	0.0011
3	0.0510	0.0301	0.0169	0.0091	0.0047	0.0024	0.0012
2	0.0706	0.0407	0.0218	0.0110	0.0053	0.0025	0.0012
1	0.1076	0.0567	0.0269	0.0121	0.0052	0.0023	0.0010
0	0.2067	0.0743	0.0267	0.0096	0.0035	0.0013	0.0005
	0	1	2	3	4	5	6

Table 3: The state probabilities after normalization

A decomposition algorithm has been developed to solve the state transition lattice of Type A structure. By dividing the lattice into several solvable subsets which can be solved sequentially, a large amount of computation time can be saved.

ACKNOWLEDGEMENT

The authors would like to acknowledge useful discussions with C.M. Manikopolous and T.G. Robertazzi, and, especially, Dr. Robertazzi's patient review of the whole paper.

REFERENCES

1. F. Baskett, K.M. Chandy, R.R. Muntz and F. Palacios, Open, Closed and Mixed Networks of ueues with Different Classes of Customers ", J. ACM, vol. 22, no. 2, April 1975.
2. J.P. Buzen, Computational Algorithms for Closed ueueing Networks with Exponential Servers ", Comm. ACM, vol. 16, no. 9, Sept. 1973.
3. W.J. Gordon and G.J. Newell, Closed ueueing Systems with Exponential Servers ", Operations Research, 15, March 1967.
4. J.R. Jackson, Networks of Waiting Lines ", Operations Research, 5, 1957.
5. L. Kleinrock, ueueing Systems, Vol.I: Theory ", Wiley-Interscience, New York, 1975.
6. A.A. Lazar and T.G. Robertazzi, The Geometry of Lattices for Markovian ueueing Networks ", Columbia University technical report, 1984. Available upon request.
7. T.G. Robertazzi, Automatic Decomposition of Certain Non-Product Form Protocol Models ", CEAS Technical Report 504, March 29, 1988. Accepted by IEEE Transactions on Communications.
8. I.Y. Wang and T.G. Robertazzi, Recursive Computation of Steady State Probabilities of Non-Product Form ueueing Models Associated with Computer Network Protocols " , 30th Midwest Symposium on Circuits and Systems, Syracuse, N.Y., Aug. 1987.
9. I.Y. Wang and T.G. Robertazzi, The Probability Flux Circulation of Certain Communication and Computation State Models ", Proc. of the 1986 Conf. on Info. Sci and Syst., Princeton N.J., 1984.
10. C.K. Hong, A Decomposition Algorithm for the State Transition Lattices of Certain Non-product ueueing Protocols ", master thesis, NJIT, 1989.

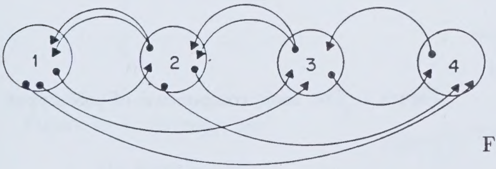


Figure 1: The basic structure of Type A

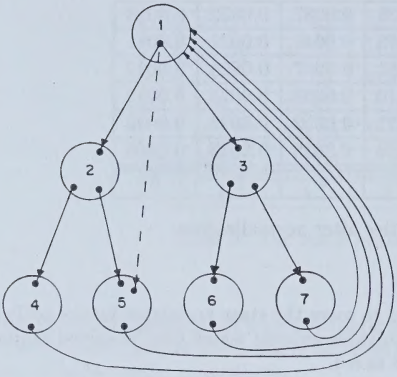


Figure 2: The basic structure of Type B

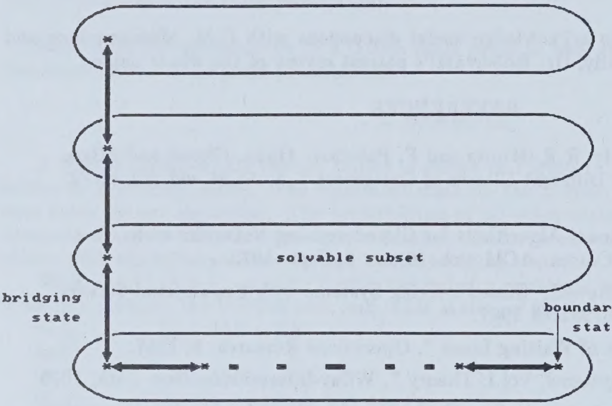

solvable subset
bridging state
boundary state

Figure 3: The regular structure of transition lattice

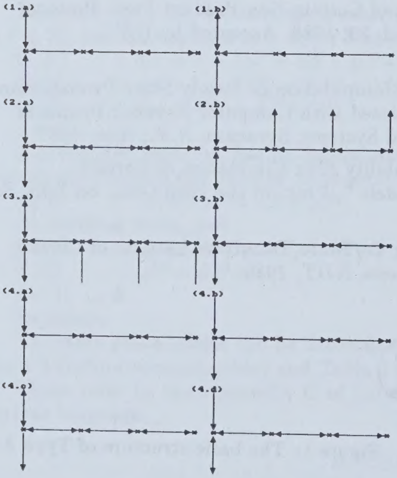

(1.a)
(1.b)
(2.a)
(2.b)
(3.a)
(3.b)
(4.a)
(4.b)
(4.c)
(4.d)

Figure 4: The basic components of prototypes

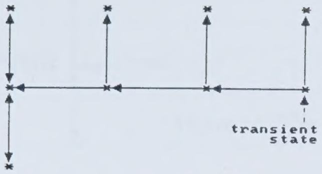

transient state

Figure 5: A basic structure of disqualified prototypes

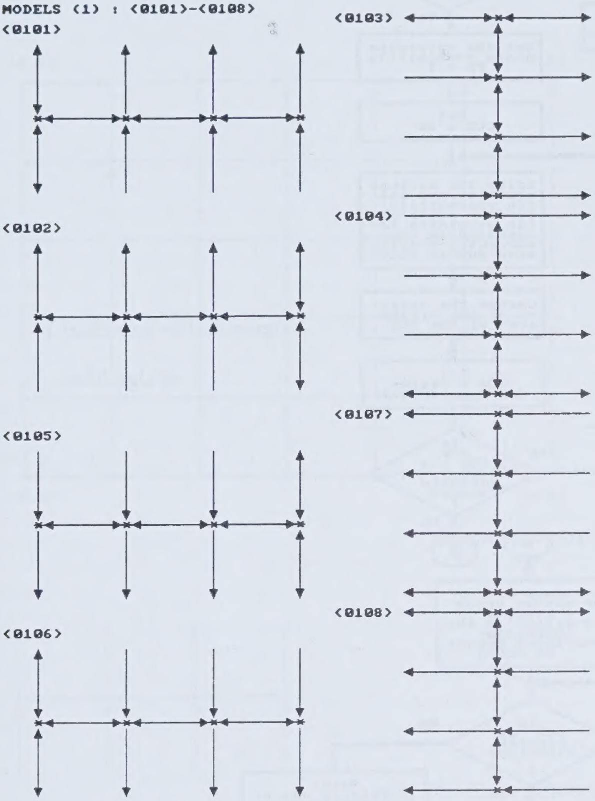

MODELS (1) : <0101>-<0108>
<0101>
<0102>
<0105>
<0106>
<0103>
<0104>
<0107>
<0108>

Figure 6: An example of 0101-model rotation

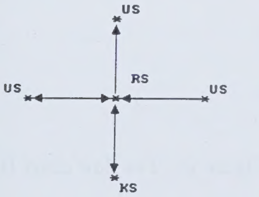

US
US
RS
US
KS

Figure 7: An example for the state vector

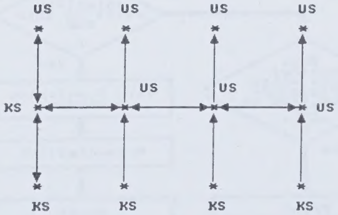

US
US
US
US
US
US
KS
US
KS
KS
KS
KS

Figure 8: The first one of the twenty four prototypes

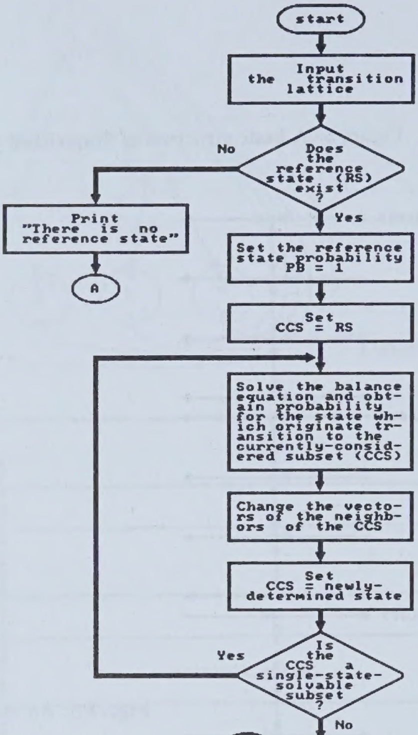

start
Input the transition lattice
Does the reference state (RS) exist ?
No
Print "There is no reference state"
A
Yes
Set the reference state probability PB = 1
Set CCS = RS
Solve the balance equation and obtain probability for the state which originate transition to the currently-considered subset (CCS)
Change the vectors of the neighbors of the CCS
Set CCS = newly-determined state
Is the CCS a single-state-solvable subset ?
Yes
No
B

Figure 9: The flow chart I of algorithm

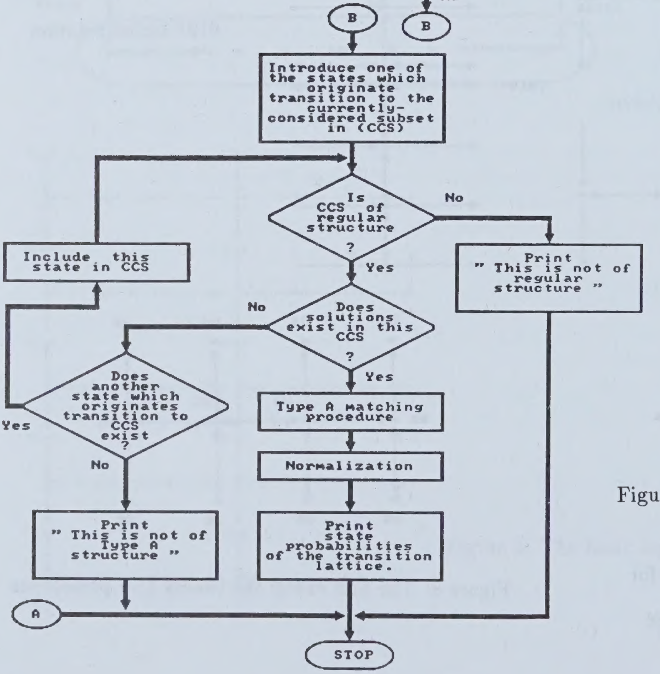

B
Introduce one of the states which originate transition to the currently-considered subset in (CCS)
Is CCS of regular structure ?
No
Print " This is not of regular structure "
Yes
Include this state in CCS
Does solutions exist in this CCS ?
No
Does another state which originates transition to CCS exist ?
Yes
No
Print " This is not of Type A structure "
Yes
Type A matching procedure
Normalization
Print state probabilities of the transition lattice.
A
STOP

Figure 10: The flow chart II of algorithm

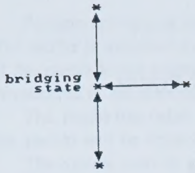

bridging
state

Figure 11: The basic structure of transitions linked with bridging state

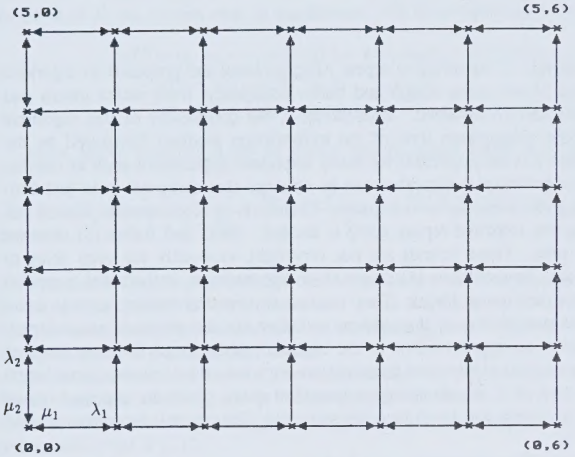

(5,0)
(5,6)
λ_2
μ_2
μ_1
λ_1
(0,0)
(0,6)

Figure 12: A typical example of Type A lattice

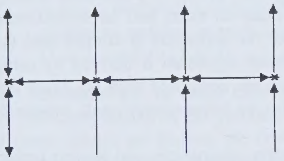


Figure 13: The 0101-prototype

A NEW ANALYSIS OF THE SELECTIVE-REPEAT ARQ PROTOCOL

SHOU-JEN WEY
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

1 INTRODUCTION

Konheim [2] presented a queueing analysis of the selective repeat ARQ protocol and proposed an algorithm for computing probability distributions of the queue length and buffer occupancy, from which means and standard deviations of these measures can be obtained. Unfortunately, the complexity of this algorithm grows exponentially with the round-trip propagation time of the transmission medium (measured by the packet transmission time), and therefore it is not applicable for many important applications such as satellite systems. It seems quite unlikely that a polynomial algorithm can be obtained that computes even just first-order statistics of queue length, waiting time, and buffer occupancy. Therefore, as Konheim has pointed out, an approximate method for analyzing the selective repeat ARQ is needed. Saeki and Rubin [3] obtained upper and lower bounds for waiting time. These bounds are not very tight, especially for large delay or heavy traffic. Recently, Anagnostou and Protonotarios [1] proposed an approximate method that computes the expected waiting time and the expected queue length. They assume that retransmission requests arrive at the transmitter independently of the past history of the system, and they use the expected queue length (waiting time) of the simplified model as an approximation to the expected queue length (waiting time) of the original system. For a given packet arrival process and transmission-error rate, the expected queue length of the approximate model is independent of d, the round-trip propagation delay, while the expected queue length of the original selective repeat system is known to be a function of d. That means, a constant is used to approximate a non-constant function.

In this paper, we propose an exact and an approximate method for computing means and variances of queue length, waiting time, and buffer occupancy, assuming a number-of-packets-per-slot arrival process. We first obtain exact formulas for performance measures such as expected queue length, expected waiting time, and their variances. These formulas are expressed in terms of primitive parameters such as the probability of an idle channel at time i given that it is idle at time j. Calculating such primitive parameters exactly or approximately and substituting them into the exact formulas, we obtain an exact and an approximate evaluation of the selective repeat ARQ protocol. In our exact method, the algorithm is simpler and the progation delay allowed is larger than previous studies. The approximate algorithm is derived by using branching process theory. Finally, the results of our approximate method are compared with the exact and simulation results, and for all cases we have observed, approximation errors are impressively small, especially when the system is under heavy load.

We are interested in the performance measures like queue lengths, buffer occupancies and waiting times.(for their definitions, see [4].) we assume , as in the literature, that the return channel is error-free so that the correctness of the response from the receiver can always be relied on.

2 THE ALGORITHM

We start with the assumptions of our model. Time is divided into intervals, called slots, of equal duration. As is typical of a slot system, the slot duration is equal to the length of a packet. (All packets are of the same length.) Without loss of generality, we assume the slot length to be 1, i.e, the transmission of a packet takes exactly one unit of time. We assume further that all packet transmissions start at slot boundaries.

The number of packets arriving at the transmitter during the interval $[t-1, t)$ is a random variable A_t. For simplicity, we assume that the arrivals of the packets all occur at time t, and the arrival process $\{A_t\}$ is a sequence of independently identically distributed (iid) random variables with common generating function $A(z)$ whose first two (factorial) moments are m_1, m_2 respectively.

Packets arriving at the transmitter are stored in the buffer until they are transmitted and acknowledged. The buffer is assumed to have unlimited capacity. Packets are transmitted on a first-come-first-served basis. If the queue is not empty, the packet at the front of the queue is transmitted in the next slot unless a NAK is received in the previous slot. In that case, the associated packet is retransmitted instead.

The round-trip delay is assumed to be d. Thus, for a packet transmitted at time t^+, an ACK/NAK for the packet will be received in the interval $[t+d-1, t+d)$.

The *system state* at time t is a $(d+1)$-tuple

$$S_t = (Q_t, U_{1,t}, U_{2,t}, \cdots, U_{d,t})$$

where Q_t is the number of packets in the queue at t^+ and $U_{i,t} = 1$ if a packet is transmitted (or retransmitted) at time $t-i+1$; and $U_{i,t} = 0$, otherwise. Let $P(y, z_1, z_2, \cdots, z_d) = E(y^Q z_1^{U_1} z_2^{U_2} \ldots z_d^{U_d})$ be the generating function of S, the system state in equilibrium. We have following functional equation:

$$\begin{aligned} &yP(y, z_1, z_2, \cdots, z_d) - ((1-e) + ey)z_1 A(y) P(y, z_2, \cdots, z_d, 1) = \\ &\quad e(1-y)z_1 A(y) P(y, z_2, \cdots, z_d, 0) \\ &\quad + a_0(1-e)(y-z_1)P(0, z_2, \cdots, z_d, 1) + a_0 e(y-z_1)P(0, z_2, \cdots, z_d, 0) \end{aligned}$$

From above eqaution, it can be shown that the queue length is given by

$$Q'(1) = \frac{1}{1-e-m_1}\{\frac{m_2}{2} + \frac{e}{1-e}m_1^2 + \frac{e(1-e-m_1)^2}{1-e}(d-1) - e(1-e)\sum_{i=2}^{d} g_{1,i}\}$$

The parameters $g_{1,i}$'s play central roles in our analysis. Roughly speaking, $g_{1,i}$ is the probability of the event that the channel is idle at time t and it will be idle again at time $t+i-1$ for sufficiently large t.Our analysis is centered on the calculation of these parameters. We give two methods, one is exact and the other is approximate, to evaluate their values.

In the exact method, first we give a systematic way to generate the coiefficients of a system of $2^{d-1}-1$ linear equations involving the primitive parameters; then solve these equations by Gauss elimination or Gauss-Siedel iteration. Although the complexity of this algorithm increase exponetially with d. It can be solved easily for $d \leq 12$.

For the approximate method, we make the assumption that the probability that a NAK is received by the transmitter in a slot is a constant ρe, where ρ is the utilization of the channel, which is equal to $\frac{m_1}{1-e}$.

The calculations of the primitive parameters is reduced to the problem of finding the distribution of busy periods of following queueing system:

> A single server queueing system: the service time for each customer is one, and the number of customers arriving in each time slot is $A + I$, where I is a random varible such that $\Pr\{I = 1\} = \rho e$ and $\Pr\{I = 0\} = 1 - \rho e$.

The calculation can be easily carried out by a recurrsive procedure.

The computation of the variance of queue length is very simular to the above. Once mean and variance of queue length are known, the other parameters, like buffer occupancy and waiting time, can be derived from them after tedious algebraic manipulations. For details, see [4].

References

[1] Miltiades E. Anagnostou and Emmanuel N. Protonotarios, "Performance Analysis of The Selective Repeat ARQ Protocol," *IEEE Trans. Commun.*, Vol. 34, No.2,Feb 1986, pp. 127-135.

[2] Alan G. Konheim , "A Queueing Analysis of Two ARQ Protocols", *IEEE Trans. Commun.*, Vol. 28, No.7, April 1980, pp.1004-11014 .

[3] Billy H. Saeki and Izhak Rubin, "An Analysis of a TDMA Channel Using Stop-and-Wait Block, and Selective-and-Repeat ARQ Error Control", *IEEE Trans. Commun.*, Vol. 30, No.5,May 1982, pp.1162-1173.

[4] Shou-jen Wey, " A New Analysis of the Selective-Repeat ARQ Protocol", *Technical Report OSU-CIS*

INPUT BUFFER REQUIREMENTS FOR ROUND ROBIN POLLING SYSTEMS

GALEN SASAKI†
Department of Electrical and Computer Engineering, The University of Texas, Austin, Texas 78712-1084

ABSTRACT

Recently, Cidon, Gopal, Grover, and Sidi proposed a model for a packet switching node when its internal structure is a work-conserving, packet-integral polling system. They provided upper and lower bounds on the minimum memory requirements to guarantee no packet loss at input buffers. For the special case of round robin polling and exhaustive service, we provide improved bounds. In addition, we extend the results to a more general polling system using a traffic characterization introduced by Cruz, and present an alternative to exhaustive service that has better memory requirements.

1. INTRODUCTION

The system considered in this paper is a packet switch, which consists of N input links, a buffer for each input link, and a high-capacity broadcast link. The input links are labeled $0, 1, ..., N-1$. Packets that arrive at an input link are stored at the buffer for the link and will eventually be transmitted on the broadcast link. The transmission capacity of each input link is S, the maximum packet length is P, and the transmission capacity of the broadcast link is S_s. We will allow fractions of bits to be stored, which should be a good approximation if typical packet lengths are long. Also, the system is initially empty of bits.

Let $R_i(t)$ (resp., $P_i(t)$) be the rate at which data is flowing (resp., the number of bits buffered) at input link i at time t. (Note that $R_i(t) \in \{0, S\}$.) We call the set $\{R_i(t) : 0 \leq i \leq N-1\}$ the *input traffic*. The following characterization of data flow was proposed by Cruz [1]. For nonnegative values σ and ρ, we say that the input traffic is (σ, ρ) if each $R_i(t)$ satisfies $\int_x^y R_i(t)dt \leq \sigma + \rho(y-x)$ for any $x \leq y$. The value ρ is an upper bound on the average value of $R_i(t)$ and σ is an upper bound on the *burstiness* of $R_i(t)$. Let $I_{[\;]}$ be the indicator function (i.e., $I_{[A]}$ equals 1 (if A is true) or 0 (if A is not true)), and let M_i be the set of times that

---

† Research supported by the Texas Advanced Research Program under project number 1643.

the broadcast link serves input link i. Then the following set of inequalities is true for all $t' \leq t$,

$$P_i(t') + \int_{t'}^{t} R_i(\tau) - S_s I_{[\tau \in M_i]} d\tau = P_i(t) \geq 0.$$

Thus, all bits transmitted by the broadcast link are no longer stored.

The packet switch can be viewed as a queueing system, where the broadcast link is a single server (serving at a rate S_s), packets are customers, and the input-link buffers hold the waiting customers. An arbitration policy determines the schedule of service of the packets. The following assumptions on the arbitration policy will hold throughout the paper. The arbitration policy is *packet integral*, which means that the broadcast link will start transmitting a packet only after the packet has been completely received by an input link buffer and will transmit the packet without interruption. The arbitration policy is also *work conserving*, which means that the broadcast link will be idle only if there are no completely received packets to be transmitted.

Let $P^*(N)$ be the supremum, over all (σ, ρ) input traffic, of $\sup_{t \geq 0} \max_i P_i(t)$; and let $P^* = \limsup_N P^*(N)$. If the input-link buffer is of size at least $P^*(N)$ then there will be no blocking at the input links. In Sections 2 and 3 we will provide upper and lower bounds for P^*. In particular, these bounds are for the round robin, exhaustive service arbitration policy, although the lower bound in Section 3 is for the special case of $\sigma = 0$ and $\rho = S$. We assume the polling is in the order of $0, 1, 2, ..., N-1$, and delays between polling links are negligible. Since the analysis is asymptotic in N, we also assume that there are constants α and β such that $\alpha = NS/S_s$ and $\beta = N\rho/S_s$. We will only consider the case $\beta \leq 1$, because if $\beta > 1$ then the broadcast link may not support the incoming data flow.

This packet switch model is a generalization of the one introduced in [2] (In that paper, $\sigma = 0$ and $\rho = S$.) They provided upper and lower bounds for P^* for the round robin, exhaustive service arbitration policy. (Recall that exhaustive service implies that once the broadcast link starts service of that link it continues service until there are no packets that can be immediately transmitted.) The bounds of Sections 2 and 3 are improvements over the ones in [2]. In the Section 4, we give an alternative service policy to exhaustive service that leads to better memory requirements under round robin polling. Final remarks are given in Section 5.

We finish the introduction with a result used in subsequent sections. Let $D = \sigma + \rho P/S$. (The following theorem is proven in [2] for the special case of $\sigma = 0$ and $\rho = S$.)

Theorem 1. For any packet integral, work-conserving arbitration policy and input traffic that is (σ, ρ), the total number of bits queued at all input links is at most ND.

Proof. The total number of bits queued at time t is

$$\int_{t'}^{t} \sum_{i=0}^{N-1} R_i(\tau)d\tau - S_s(t-t') + \sum_{i=0}^{N-1} P_i(t'), \tag{1}$$

where t' is the last time the broadcast link was not transmitting. Since the system is work conserving, at time t' none of the input-link buffers had a complete packet. Thus $P_i(t') \leq P$. Also, note that

$$\int_{t'}^{t} R_i(\tau)d\tau + P_i(t') = \int_{t'-P_i(t')/S}^{t} R_i(\tau)d\tau$$

$$\leq \sigma + \rho(t - t' + P_i(t')/S) \leq \sigma + \rho(t-t') + \rho P/S.$$

Thus, Expression (1) is at most $N(\sigma + \rho P/S)$ $+(N\rho - S_s)(t-t')$. Recall, $\beta \leq 1$, therefore $N\rho - S_s \leq 0$ and Expression (1) is at most $N(\sigma + \rho P/S)$.

□

2. AN UPPER BOUND FOR P* UNDER ROUND ROBIN POLLING AND EXHAUSTIVE SERVICE

In this section we will consider round robin polling and exhaustive service. Let $X_i(j)$ (resp., $Y_i(j)$) to be the time that the broadcast link starts (resp., finishes) the jth service of link i. (Note that $X_0(j) = Y_{N-1}(j-1)$ and $X_i(j) = Y_{i-1}(j)$ for $i > 0$.) For link i, interval $(Y_i(j), X_i(j+1)]$ is called the *jth vacation* interval and $(X_i(j), Y_i(j)]$ is called the *jth busy* interval. Note that $P_i(t)$ is nondecreasing on vacation intervals of i, nonincreasing on busy intervals of i, and is continuous (since the transmission rates are finite). Thus, $P_i(t) \leq \sup_j P(X_i(j))$. Let $v(i)$ equal the supremum of $X_i(j+1) - Y_i(j)$ (i.e., the length of the jth vacation) over all j and (σ, ρ) input traffic $\{R_i(t) : 0 \leq i \leq N-1\}$; let $v = \max_{0 \leq i \leq N-1} v(i)$. If V is an upper bound for $v(i)$ then $\sigma + \rho(V + P/S)$ is an upper bound for $P_i(t)$, since

$$P_i(X_i(j)) = \int_{Y_i(j-1)}^{X_i(j)} R_i(\tau)d\tau + P_i(Y_i(j-1)) = \int_{Y_i(j-1)-P_i(Y_i(j-1))/S}^{X_i(j)} R_i(\tau)d\tau$$

$$\leq \sigma + \rho[X_i(j) - Y_i(j-1) + P_i(Y_i(j-1))/S] \leq \sigma + \rho(V + P/S).$$

Let $B(V) = \sigma + \rho(V + P/S)$. It also follows that if V is an upper bound for v then $B(V) \geq P^*(N)$.

We now turn to finding an upper bound for $X_{N-1}(j) - Y_{N-1}(j-1)$ (i.e., the length of the $j-1th$ vacation of link $N-1$), which will also be an upper bound for v. For brevity, let $T_i = Y_i(j)$, $T'_i = Y_i(j-1)$, $T_{-1} = Y_{N-1}(j-1)$, $K_i = P_i(T_{-1})$, and $\Delta_i = T_i - T_{-1}$. Notice that $\Delta_{N-2} = X_{N-1}(j) - Y_{N-1}(j-1)$, the length of the $j-1th$ vacation of link $N-1$. Then

$$\int_{T_{-1}}^{T_i} R_i(\tau)d\tau - S_s(T_i - T_{i-1}) + K_i \geq 0$$

$$\sigma + \rho\Delta_i - S_s(\Delta_i - \Delta_{i-1}) + K_i \geq 0$$

$$\Delta_i \leq \frac{\sigma + K_i}{S_s - \rho} + \frac{S_s}{S_s - \rho}\Delta_{i-1}.$$

This implies, $\Delta_i \leq \sum_{m=0}^{i}[(\sigma + K_m)/(S_s - \rho)][S_s/(S_s-\rho)]^{i-m}$.

Since $\sum_{m=0}^{N-1} K_m \leq DN$ (from Theorem 1) and $S_s/(S_s - \rho) \geq 1$, it follows that an upper bound for Δ_{N-2} is

$$V_0 = \sum_{m=0}^{N-2} \frac{\sigma}{S_s - \rho}(\frac{S_s}{S_s - \rho})^{N-2-m} + \frac{DN}{S_s - \rho}(\frac{S_s}{S_s - \rho})^{N-2}.$$

Note that V_0 is not dependent on j so it is also an upper bound for $v(N-1)$. One can similarly verify that V_0 is a bound for $v(i)$ for all i such that $0 \leq i \leq N-1$. As N goes to infinity, V_0 converges to $(\sigma/\rho)(e^{\beta}(\beta+1) - 1)$ $+\beta e^{\beta}P/S$. (The upper bound V_0 is derived in [2] for the special case of $\sigma = 0$ and $\rho = S$.)

Suppose V is an upper bound for v. Note that

$$K_m = \int_{T'_m}^{T_{-1}} R_m(\tau)d\tau + P_m(T'_m) \leq \sigma + \rho(T_{-1} - T'_m + P/S). \tag{2}$$

Since $T_{m-1} - T'_m =$ (length of the $j-1th$ vacation for link m) $\leq V$ and $T_{m-1} - T_{-1} \geq \sum_{j=0}^{m-1} K_j/S_s$,

$$\sum_{j=0}^{m-1} K_j/S_s + T_{-1} - T'_m \leq T_{m-1} - T_{-1} + T_{-1} - T'_m \leq T_{m-1} - T'_m \leq V.$$

Thus, $T_{-1} - T'_m \leq V - \sum_{j=0}^{m-1} K_j/S_s$, which together with Inequality (2) imply $K_m \leq B(V) - \rho\sum_{j=0}^{m-1} K_j/S_s$. An upper bound for Δ_{N-2} is the solution to the following problem

$$\max \sum_{m=0}^{N-2} \frac{\sigma + K_m}{S_s - \rho}(\frac{S_s}{S_s - \rho})^{N-2-m} \tag{Problem 1}$$

subject to $\sum_{m=0}^{N-1} K_m \leq DN$ and for all m,

$$0 \leq K_m \leq B(V) - \rho \sum_{j=0}^{m-1} K_j / S_s.$$

Note that the solution is also an upper bound for v.

Lemma 1. Suppose $V \geq v$ and J satisfies

$$B(V) \sum_{i=0}^{J} (1 - \rho/S_s)^i \leq DN < B(V) \sum_{i=0}^{J+1} (1 - \rho/S_s)^i$$

Then the optimal solution to Problem 1 is

$$K_i^* = \begin{cases} B(V)(1 - \rho/S_s)^i, & \text{if } i \leq J; \\ DN - B(V) \sum_{i=0}^{J} (1 - \rho/S_s)^i, & \text{if } i = J + 1; \\ 0, & i > J + 1. \end{cases}$$

Proof. Suppose that there is a j such that $K_j < B(V) - (\rho/S_s) \sum_{i=0}^{j-1} K_i$, $K_i = 0$ for all $i > j$, and $\sum_{i=0}^{N-1} K_i < DN$. Then $\{K_i\}$ is not optimal since the following feasible solution has larger value:

$$K_i' = \begin{cases} K_i, & \text{if } i \neq j; \\ K_i + \delta, & \text{if } i = j; \end{cases}$$

where $\delta = \min\{B(V) - (\rho/S_s) \sum_{i=0}^{j-1} K_i - K_j, DN - \sum_{i=0}^{N-1} K_i\}$.

Suppose there is a j such that $K_j < B(V) - (\rho/S_s) \sum_{i=0}^{j-1} K_i$ and $K_i > 0$ for some $i > j$. Then $\{K_i : 0 \leq i \leq N-1\}$ is not optimal because the following feasible solution has larger value. Let $Q = \{i : i > j \text{ and } K_i > 0\}$, $\delta' = \min\{B(V) - (\rho/S_s) \sum_{i=0}^{j-1} K_i - K_j, \min_{i \in Q} K_i\}$, and the feasible solution be

$$K_i' = \begin{cases} K_i, & \text{if } i < j; \\ K_i + \delta', & \text{if } i = j; \\ \max\{0, K_i - \delta'/|Q|\}, & \text{if } i > j\ ; \end{cases}$$

where $|Q|$ is the cardinality of Q. The solution is feasible since for $m > j$ and $K_m > 0$,

$$B(V) - \frac{\rho}{S_s} \sum_{i=0}^{m-1} K_i' = B(V) - \frac{\rho}{S_s} \sum_{i \neq j} K_i' - \frac{\rho}{S_s}(K_j + \delta')$$

$$\geq (B(V) - \frac{\rho}{S_s} \sum_{i=0}^{m-1} K_i) - \frac{\rho}{S_s} \delta' \geq K_m - \delta' \beta / N \geq K_m'.$$

Hence, if $\{K_i : 0 \leq i \leq N-1\}$ is optimal and there is a j such that $K_j < B(V) -(\rho/S_s)\sum_{i=0}^{j-1} K_i$ then for all $i > j$, $K_i = 0$ and $\sum_{i=0}^{N-1} K_i = DN$. Note that $\{K_i^* : 0 \leq i \leq N-1\}$ is the only solution that satisfies those conditions.

□

Let $f_N(V) = \sum_{m=0}^{N-2}[(\sigma + K_m^*)/(S_s - \rho)][S_s/(S_s - \rho)]^{N-2-m}$. Then as N goes to infinity $f_N(V)$ converges to

$$f(V) = \frac{\sigma}{\rho}(e^{\beta} - 1) + \frac{e^{\beta}\beta D}{\rho}(1 - \frac{D\beta}{2B(V)}),$$

where $D = \sigma + \rho P/S$. A solution for $V = f(V)$ is

$$V^* = \frac{\rho b_1 - D + \sqrt{(D + \rho b_1)^2 - 4\rho b_2}}{2\rho}$$

$$b_1 = \frac{\sigma}{\rho}(e^{\beta} - 1) + \frac{\beta e^{\beta} D}{\rho} \qquad b_2 = \frac{\beta^2 D^2 e^{\beta}}{2\rho}$$

Let the sequence $V_0, V_1, ...$ be such that V_0 is defined above and $V_j = f_N(V_{j-1})$. Then each V_j is an upper bound for v, because V_0 is an upper bound for v. It can be shown that (i) f_N converges uniformly to f over any finite interval $[a, b]$ where $a > V^*$ and (ii) $V - f(V)$ is strictly positive and increasing on (V^*, ∞). This implies that for any $\epsilon > 0$, there is an N' such that $V^* + \epsilon \geq v$ and $B(V^* + \epsilon) \geq P^*(N)$ for all $N \geq N'$.

Theorem 2. For round robin polling, exhaustive service, and (σ, ρ) input traffic $P^* \leq B(V^*)$.

Finally, consider the special case: $\sigma = 0, \rho = S$, and $\beta = 1$. Then $P^* \leq 3.307P$. This is a slight improvement over the bound in [2], which is $3.350P$. Other advantages of our bound is that it is for a more general polling system model and it is a closed-form expression. (The bound in [2] was found numerically).

3. A LOWER BOUND FOR P* UNDER ROUND ROBIN POLLING AND EXHAUSTIVE SERVICE

In this section we consider the exhaustive service policy and provide lower bounds for P^*. Assume that the polling is round robin and that link 0 is the first to be polled. A particular input traffic $\{R_i(t) : 0 \leq i \leq N-1\}$ will be defined and used to get a lower bound for P^*. Let τ_1 equal $\sigma/(S - \rho)$ (if $\rho < S$) or ∞ (if $\rho = S$), $\epsilon > 0$ be a very small constant, and H_k be the interval $(\tau_1 + \epsilon k, \tau_1 + \epsilon k + \epsilon\rho/S]$. For each i,

$$R_i(t) = \begin{cases} 0, & \text{if } t < 0; \\ S, & \text{if } 0 \le t \le \tau_1; \\ S\sum_{k=1}^{\infty} I_{[t\in H_k]}, & \text{if } t > \tau_1 \ ; \end{cases}$$

where $\epsilon > 0$ is very small. For small ϵ, $R_i(t)$ is well approximated by

$$R(t) = \begin{cases} 0, & \text{if } t < 0; \\ S, & \text{if } 0 \le t \le \tau_1; \\ \rho, & \text{otherwise .} \end{cases}$$

Note that $\int_0^t R(\tau)d\tau = D + t\rho$. We also assume that for all links, the first packet to be received is of length P and subsequent packets are of infinitesimal length. Thus, the first packet is completely received at time P/S and $X_0(1) = P/S$.

Let $\pi_i^j = Y_i(j) - X_0(j)$ and $\pi_i^{-1} = 0$. We now turn to computing π_i^1. Note that

$$\int_0^{Y_i(1)} R(\tau)d\tau - S_s(Y_i(1) - X_0(1)) = 0$$

$$P + \int_{X_0(1)}^{Y_i(1)} R(\tau)d\tau - S_s(\pi_i^1 - \pi_{i-1}^1) = 0.$$

Case 1. Suppose $Y_i(1) \le \tau_1$:

$$P + S\pi_i^1 - S_s(\pi_i^1 - \pi_{i-1}^1) = 0$$

$$\pi_i^1 = \frac{P}{S_s - S} + \frac{S_s\pi_{i-1}^1}{S_s - S} = \sum_{j=0}^{i} \frac{P}{S_s - S}\left(\frac{S_s}{S_s - S}\right)^j.$$

If N is large and $\omega = i/N$ then

$$\pi_i^1 \approx \frac{NP}{S_s}\frac{1}{N}\sum_{j=0}^{\omega N}\left(\frac{1}{1-\alpha/N}\right)^j \approx \frac{\alpha}{S}P\int_0^{\omega} e^{\alpha x}dx = \frac{P}{S}[e^{\alpha\omega} - 1].$$

Case 2. Let τ_1 be in the first busy interval of input link L:

$$\pi_L^1 = \frac{P - (S-\rho)(Y_j(1) - \tau_1)}{S_s - S} + \frac{S_s}{S_s - S}\sum_{i=0}^{L-1}\frac{P}{S_s - S}\left(\frac{S_s}{S_s - S}\right)^j.$$

Let $\lambda = L/N$. If N is large then $\pi_L^1 \approx \frac{\sigma}{S-\rho} - \frac{P}{S}$. Therefore,

$$\frac{P}{S}[e^{\alpha\lambda} - 1] \approx \frac{\sigma}{S-\rho} - \frac{P}{S} \quad \text{and} \quad e^{-\alpha\lambda}\frac{\sigma}{S-\rho} \approx \frac{P}{S}.$$

Case 3. Suppose $i > L$: By similar calculations,

$$\pi_i^1 = \frac{P}{S_s - \rho} \sum_{k=0}^{i-L-1} (\frac{S_s}{S - \rho})^k + \pi_L^1 (\frac{S_s}{S_s - \rho})^{i-L}.$$

If N is large and $\omega = i/N$ then

$$\pi_i^1 \approx e^{\beta(\omega - \lambda)} \frac{\sigma}{S - \rho} - \frac{P}{S} = e^{\beta\omega} (e^{-\alpha\lambda} \frac{\sigma}{S - \rho})^{\beta/\alpha} (\frac{\sigma}{S - \rho})^{1 - \beta/\alpha} - \frac{P}{S}$$

$$= e^{\beta\omega} (\frac{P}{S})^{\beta/\alpha} (\frac{\sigma}{S - \rho})^{1 - \beta/\alpha} - \frac{P}{S} = \frac{P}{S} [e^{\beta\omega} (\frac{S}{\rho} \frac{\sigma}{S - \rho})^{1 - \beta/\alpha} - 1].$$

Note that for exhaustive service and any polling strategy, $\int_0^{Y_{N-2}(1)} R(\tau) d\tau = \sigma + \rho(\pi_{N-1}^1 + \frac{P}{S}) = D + \rho\pi_{N-1}^1$ is a lower bound for $P^*(N)$, since $R_i(t)$ is identical over i.

Theorem 3. For exhaustive service, any polling strategy, and (σ, ρ) input traffic, a lower bound for P^* is

$$D + \begin{cases} \rho \frac{P}{S}(e^\alpha - 1), & \text{if } \frac{P}{S}(e^\alpha - 1) \geq \frac{\sigma}{S-\rho} - \frac{P}{S}; \\ \rho \frac{P}{S}[e^\beta [\frac{S}{P} \frac{\sigma}{S-\rho}]^{1-\beta/\alpha} - 1], & \text{otherwise .} \end{cases}$$

For the special case of $\sigma = 0$ and $S = \rho$, the bound is Pe^α.

We now turn to round robin polling and the special case of $\sigma = 0$ and $S = \rho$. For this case, $\tau_1 = \infty$. Let $\Phi_i = P_i(X_0(2))$. Then $\Phi_i = S(X_0(2) - Y_i(1)) = S(Y_{N-1}(1) - Y_i(1)) = S(\pi_{N-1}^1 - \pi_i^1)$. For large N, $\Phi_i \approx k(i/N)$, where $k(r) = P(e^\beta - e^{\beta r})$. Next, $\Phi_i + S\pi_i^2 - S_s(\pi_i^2 - \pi_{i-1}^2) = 0$, $\pi_i^2 = [\Phi_i/(S_s - S)]\pi_{i-1}^2$, and $\pi_i^2 = \sum_{j=0}^{i} [\Phi_j/(S_s - S)][S_s/(S_s - S)]^{i-j}$. For large N,

$$\pi_i^2 \approx \int_0^{i/N} k(r) \frac{\beta}{S} exp(\beta[(i/N) - r]) dr = g(i/N),$$

where $g(r) = \frac{P}{S}[e^\beta(e^{\beta r} - 1) - re^{\beta r}]$. Then the first vacation of link i has length $\pi_{i-1}^2 + Y_{N-1}(1) - Y_i(1) \approx g((i-1)/N) + k(i/N)/S \approx (P/S)e^{\beta r}(e^\beta - r - 1)$, where $r = i/N$. The function $e^{\beta r}(e^\beta - r - 1)$ has a maximum value of $(1/\beta)exp(\beta e^\beta - \beta - 1)$, when r is equal to $\theta = e^\beta - 1 - 1/\beta$.

Let J be the integer part of θN. Recall that π_{J-1}^1 $(= X_{J-1}(1) - X_0(1))$ converges to $(P/S)(e^{\beta\theta} - 1)$. Change the data flow to input link J from $R_J(t)$ to $R'_J(t)$, where $R'_J(t) = R_J(t - (P/S)(e^{\beta\theta} - 1) + \epsilon)$ and $\epsilon > 0$ is a small number. For sufficiently large N, $\pi_{J-1}^2 + Y_{N-1}(1) - Y_i(1)$ is still approximately $(P/S)e^{\beta\theta}(e^\beta - \theta - 1)$. Also $P_J(X_J(1))$ is less than and approximately equal to P and hence

J does not transmit at time $X_J(1)$. It follows that $P_J(X_J(2))$ is approximately $Pe^{\beta\theta}(e^{\beta} - \theta - 1) + P$.

Theorem 4. For round robin polling, exhaustive service, and (σ, ρ) input traffic where $\sigma = 0$, and $\rho = S$, $P^* \geq P[\frac{1}{\beta} exp(\beta e^{\beta} - \beta - 1) + 1]$.

For $\beta = 1$, $P^* \geq 3.051P$.

4. AN ALTERNATIVE TO EXHAUSTIVE SERVICE

The following alternative to exhaustive service affords better bounds for round robin polling, which is assumed throughout the section. The alternative service requires that input links can broadcast control messages to other input links with negligible propagation and processing delays. A contiguous set of busy intervals is called a *busy frame.* The first input link to be polled in the frame is called the *synchronizer* of the frame. Every time the synchronizer is polled during the frame, it broadcasts a control message to indicate the start of a *cycle. Cycles* are defined to be the intervals within the frame that are partitioned by the transmission times of the messages. When an input link is polled during a cycle, it sends only the packets that have been completely received at the start of the cycle. We call this service policy *real-time gated service.*

Let C_i (resp., E_i) be time of the start (resp., end) of the *ith* cycle. If the cycle is the first in a frame then $P_j(C_i)$ is at most P, since the system is work conserving. Let $\Lambda_j(i)$ be the length of any incompletely received packet at input link j at time C_i. If cycle i is not the first in a frame then

$$P_j(C_i) \leq \Lambda_j(C_{i-1}) + \int_{C_{i-1}}^{E_{i-1}} R_j(\tau)d\tau = \int_{C_{i-1}-\Lambda_j(i-1)/S}^{E_{i-1}} R_j(\tau)d\tau$$

$$\leq \sigma + \rho(E_{i-1} - C_{i-1} + \Lambda_j(i)/S) \leq D + \rho\frac{ND}{S_s} = D(1+\alpha).$$

Note that in the last inequality we used the fact that

$$E_{i-1} - C_{i-1} \leq \frac{1}{S_s}\sum_{k=0}^{N-1} P_k(C_{i-1}) \leq ND/S_s$$

by Theorem 1.

Thus, if t is in cycle i then

$$P_j(t) = P_j(C_i) + \int_{C_i}^{t} R_j(\tau)d\tau \leq P_j(C_i) + \sigma + \rho(t - C_i)$$

$$\leq D(1+\alpha) + \sigma + \rho ND/S_s \leq \sigma + D(1+2\alpha).$$

Theorem 5. For round robin polling, real-time gated service, and (σ, ρ) input traffic, $P^* \leq \sigma + D(1 + 2\alpha)$.

Note that for the special case of $\sigma = 0$, $\rho = S$, and $\alpha = 1$, the real-time gated system requires at most $3.00P$ bits at each input link, while the exhaustive service requires at least $3.05P$ bits.

5. FINAL REMARKS

We have provided new bounds on memory requirements for a work-conserving, packet-integral switching system so that no blocking occurs at input-link buffers. Most of our results apply to the exhaustive service policy and round robin polling. Another service policy, called real-time gated, is given which has better memory requirements than exhaustive service for round robin polling. However, the improvements of real-time gated service is slight and requires additional control information overhead.

ACKNOWLEDGEMENTS

The author would like to thank Rene Cruz for stimulating discussions.

REFERENCES

[1] R. Cruz, "A calculus for network delay and a note on topologies of interconnection networks," Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Tech. Rept. UILU-ENG-87-2246, July 1987, (Ph.D. Dissertation)

[2] I. Cidon, I. Gopal, G. Grover, and M. Sidi, "Real-time packet switching: A performance analysis," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1576-1586, Dec. 1988.

Maximum Queues in M/M/1 and M/D/1 Systems

Werner Furth and Nicholas Duchon

Martin Marietta Laboratories, 1450 South Rolling Road, Baltimore, MD 21227
(301) 247-0700

Abstract: The solution for the statistical properties of the maximum queue ($Q_{max}(t)$) encountered in an M/M/1 system is presented, along with some of the qualitative and quantitative results of that solution. We show, for the utilization factor (ρ) less than one and large times, that the mean value of Q_{max} is proportional to $-\ln[\, t(1-\rho)^2 \,]/\ln[\rho]$, and that for small values of ρ, σ_m is periodic in $\ln[t]$ and is bounded. For large t and for $\rho \geq 1$, other simple approximations can be justified and are confirmed by computations. The results of Monte-Carlo simulations for the M/D/1 system are shown, and a simple method of converting $E(Q_{max})$ from the M/M/1 solution to the M/D/1 system is proposed.

1. INTRODUCTION

Many of the statistical properties of single-server systems, with quite general arrival time and service time distributions, have been studied $^{(1\text{-}4)}$. In this abbreviated paper, rather than re-examining the statistical properties of the number of items that may be in the system, we will expand on a slightly different question: if $Q_{max}(t)$ is the maximum queue that has been experienced from the start (t=0, with the system empty) to time t, what are the statistical properties of $Q_{max}(t)$? This paper is a condensation of of a longer report$^{(5)}$.

There are fundamental differences between Q(t) and $Q_{max}(t)$. As time becomes large, for $0 < \rho < 1$ (where ρ is the mean time to service/ the mean time between arrivals), all the statistics of Q(t) become independent of t. In particular, as $t \to \infty$, the mean value (ensemble or time average) of Q(t) approaches a finite number. In contrast, for all $\rho > 0$, the ensemble mean of $Q_{max}(t)$ becomes infinitely large as $t \to \infty$. In fact, $Q_{max}(t) \to \infty$ as $t \to \infty$ with probability 1.

Why then, when it has such nasty properties, is $Q_{max}(t)$of interest? The answer is that, for all processes where there are has finite limits on the capacity of queues or on the waiting time in a queue (reneging, balking), it is desirable to be able to evaluate and understand the probabilities of exceeding certain levels in a finite time period. This probability may be very large compared to that calculated from the first two moments of Q(t), using seemingly plausible factors for increasing σ_Q or E(Q) to account for the passage of time.

Since a plethora of processes can be imagined for both service and arrival times, we limited our investigation to two systems: the M/M/1 system, for which the whole distribution function for $Q_{max}(t)$ can be determined analytically and the first two moments readily evaluated; and the M/D/1 system, which was studied by Monte Carlo simulation for a range of times and ρ's. We are primarily interested in the behavior of Q_{max} for "large" times, i.e., non-dimensional times (time divided by service time; see next section) on the order of 10^3 or larger. We are also interested in the distribution functions, the mean, and the variance.

2. EQUATIONS, MAXIMUM QUEUE M/M/1 SYSTEM

Let us suppose a simple queueing system with both the intervals between arrivals and the time to service given by statistically independent exponential distributions with constant coefficients, i.e., "Poisson." In this system, there are no limits on the number of items in the queue.

At the start, at $t = 0$, the queue is empty. The non-absorption "state k" denotes the situation where exactly k items are in the queue, including any which are being processed.

Our approach for calculating the distribution and, hence, the moments of $Q_{max}(t)$, is to set up and solve the differential equations for an M/M/1 system (as described above) in state i, $i = 0,1,\dots,n+1$, where state $n+1$ is an absorption state. Initially, the system is in state 0. This definition of states implies that the probability that the system is in state $n+1$ at some time t is the probability that a queue of size n has been exceeded at some time in the process. We will call this probability $z_{n+1}(t)$. Since the probability that $Q_{max}(t)$ is equal to some integer k is equal to $z_k(t)-z_{k+1}(t)$, the probability distribution and the first few moments for $Q_{max}(t)$ can be determined. We evaluated these parameters for a variety of ρ's and times, and examined limiting cases both analytically and numerically. From these results, we have extrapolated to systems other than M/M/1.

When writing the equations, it is natural to non-dimensionalize the time T by $t = \mu T$, where for the Poisson processes, the arrival rate λ and the service rate μ are simply the inverses of the corresponding mean times. Since the equations with non-dimensionalized time contain the ratio of the rates, we will let $\rho = \lambda/\mu$ (Utilization Factor$^{(1)}$).

From these equations, we clearly see that z_{n+1} is 1 minus a linear combination of $e^{\beta_i t}$, where we determine the $n+1$ exponents β_i (which are also functions of n and ρ) by setting equal to zero the determinant of the matrix resulting from the Laplace transform of those equations. Once the β_i have been found, the coefficients multiplying the term $e^{\beta_i t}$, called c_i and also a function of n and ρ, can be readily determined from the linear algebraic equations

$$\{c\} * [[\beta_1, \beta_2, \dots \beta_n]] = \{1,0,\dots 0\}.$$

The shorthand notations used for the Vandermonde matrix $[[\beta_1, \beta_2, \dots \beta_n]]$ and the vectors $\{\dots\}$ are explained in the Notation section. These equations have the solution

$$c_i = \Pi\, [\beta_k/(\beta_k - \beta_i)]$$

where the product is taken for all k values not equal to i, and where

$$z_{n+1} = 1 - \Sigma^i\, c_i e^{\beta_i t}.$$

It should be noted that all the roots β_i are negative. In this paper, we have implicitly assumed that they are distinct (which is true) and real (also true). It should also be mentioned that even if the initial conditions are changed, the roots β_i and the subsequent equations are not changed; only the expressions for c_i are modified by changes in the vector on the right side of the linear algebraic equation above.

Taking the Laplace transform, $L\{x\}$, results in an n+1 by n+1 matrix, where the terms on three diagonals are constants (-1, $-\rho$) or linear expressions of x ($x+\rho$, $x+\rho+1$). We will call the determinant of that matrix $A_{n+1}(x)$. Since we have to calculate $A_n(x)$ for high values of n (in fact, for $n \rightarrow \infty$), it is interesting to see that the $A_n(x)$ satisfy the following recursion equation:

$$A_{n+1}(x) = (x + \rho + 1)A_n(x) - \rho A_{n-1}(x).$$

From this equation, it is immediately clear that the roots of $A_n(x) = 0$ are such that

$$-\Sigma^i\, 1/\beta_i = (1-\rho)^{-2}[n\rho - (n+1) + \rho^{-n}]$$

which is useful in obtaining approximations of c_i for large n. It turns out that the latter expression also is the average time to first exceed a level n.

Even though the above recursion equation for A_n can be solved in closed form, and in principle all the required roots can be determined, it is not easy either to see the properties of the solutions or to numerically evaluate them. However, if we let $\zeta = ½(x + \rho + 1)/\rho^{½}$ and also scale A_n, then it is easily shown that for n = 1,2,...

$$\Phi_n(\zeta) = \rho^{-½n}A_n(\zeta) = [\, U_n(\zeta) - U_{n-1}(\zeta)/\rho^{½}\,]$$

where $U_n(\zeta)$ are the Tchebychef polynomials of the second kind. Therefore, to find the zeros of the determinant, we have to find all the solutions for $0 \leq \theta < \pi$ or for θ imaginary of the following equation

$$\sin[(n+1)\theta]/\sin(\theta) = \sin[n\theta]/[\rho^{½}\sin(\theta)]$$

For the solutions (subscripted i) the desired roots are given by the various values of θ_i as follows:

$$\beta_i = 2\rho^{½}\cos(\theta_i) - (1+\rho).$$

For $\rho > 1$ or for sufficiently small n all the roots for θ are real, and there are a sufficient number of roots in the interval 0 to π, all leading to different values of β. However, since it is the largest β that will control the behavior of the sums and their behavior for large times, we now concentrate on the imaginary solution(s) for θ, if any.

There is an imaginary root, and there is at most one, if and only if $[n/(n+1)]^2 > \rho$. This imaginary root for θ results in $\zeta > 1$, which is the largest ζ. Since, for large times, we want the β_i with the smallest magnitude (β_i are all negative), the properties of the largest ζ are of interest. It can be shown that if n is large and ρ is small (at least a bit less than 1), then the largest β_i is $\simeq -(1-\rho)^2\rho^n$, and the c_i for that root $\simeq 1 - (n+1)\rho^n + n\rho^{n+1}$. This implies that for large n, small ρ, and large but not excessively large t, then in the vicinity of the mean, the probability that max queue equals k is approximately $t(1-\rho)^3\rho^k$.

There is another implication. From either the relationship above or the limiting equation for $z_k(t)$, we can reasonably expect that for large times

$$E(Q_{max}(t)) = \text{a linear function of } \{-\ln[\, t(1-\rho)^2 \,] \,/\, \ln[\, \rho \,]\}$$

with an educated guess that the slope should be close to 1. Among other things, we can then infer that the cumulative distribution for large time is given by

$$\ln\,[1 - z_n] \rightarrow -\rho^{\Delta n}$$

where $\Delta n = n - E(Q_{max}(t))$. Consequently, the moments about the mean become independent of time, at least for reasonably sized ρ.

3. RESULTS FOR M/M/1

Even with the analytical solutions on hand, it is still tedious to use them since the myriad coefficients involved have to be numerically calculated. Therefore, we present a discussion of the properties under some limiting cases, and then the results of numerical computer calculations.

Small Time

For small values of ρt, the only reasonably probable values that Q_{max} can have are 0, 1, and possibly 2. We can readily show that

$$E(Q_{max}) \simeq \sigma_m^2 \simeq \rho t$$

and that the probability distribution closely resembles a Poisson distribution.

Large Time and Large ρ

If ρ is larger than one, then considerably more arrivals than departures would be expected over a large time interval. Therefore, after some time, it would be very unlikely for the queue to be empty. We would therefore expect that the mean number of departures in a time interval then would be simply $\mu\ \Delta T$ and the mean number of arrivals would be $\lambda \Delta T$, a considerably larger number. The maximum and mean would therefore continually increase, so that $E(Q_{max})$ would be approximately $E(Q)$, and both would be approximately $(\rho\text{-}1)t$. It is also probable that $\sigma_m^2(t)$ would be about $\sigma_Q^2(t)$ and equal to approximately $(\rho+1)t$.

For this case, the probability distribution for Q_{max} would likely resemble a normal distribution. Further, it's reasonable to suppose that the mean value would be insensitive to the probability distributions of the arrival or the service, i.e., that this result would apply to G/G/1 systems.

Since $\sigma_m^2(t)$ and $\sigma_Q^2(t)$ would be zero in a D/D system, it is not unreasonable to assume that $\sigma_{[m\ \text{or}\ Q]}^2(t) \simeq ½\,[\sigma_a^2/(E(t_a))^2 + \sigma_s^2/(E(t_s))^2]\ (\rho + 1)t$. Thus for Poisson arrivals with fixed service times, both the variance of the mean and the maximum ($\sigma_Q^2(t)$ and $\sigma_m^2(t)$) decrease by a factor of 2.

Large Time and Small ρ

The case of large time and small ρ is an interesting one. As has been noted, as time becomes large, $E(Q_{max}(t))$ appears to be a linear function of $-\ln[\ t(1\text{-}\rho)^2\ ]\ /\ \ln[\ \rho\ ]$ (see below). For the variance of Q_{max}, however, some other phenomena appear.

For values of ρ less than 1 but not "too small," theory predicts and computations confirm that the variance, skewness, and the shape factors for the probability distribution become approximately constant with time.

However, the situation is different for small values of ρ. A plot of the standard deviation as a function of time and ρ, presented in Figure 1, shows the periodicity of the standard deviation with the independent variable $\ln[t]$. The period is equal to $1/\rho$, the ratio between the maximum and the minimum is roughly proportional to $\rho^{-1/2}$, and the maximum standard deviation is approximately ½.

This periodicity initially surprised us, but with the benefit of hindsight it is readily predictable:

> For very small ρ at a fixed time, clearly only a very few values of Q_{max} have a non-negligible chance of occurring. However, the mean increases with time (roughly linear with $\ln[t]$); consequently, when the mean is an integer, it is very likely (probability close to 1) that the maximum queue is equal to that integer, and very unlikely that it is equal to a number on either side of that integer. Thus, in this case, the variance of the maximum queue is small. However, as time moves on, the mean further increases, implying that the distribution becomes bimodal as the mean reaches a value of the integer plus ½. Q_{max} then has an approximately equal probability of being the previous integer or that integer plus 1. For this situation, then, the variance is about 0.25. With additional time the mean becomes equal to the next integer, and the variance is again small. Hence, the standard deviation oscillates with $\ln[t]$.

The key to this result is the dependency of $E(Q_{max})$ on time, as shown in Figure 2, a plot of the mean (scaled by the expected number of arrivals, ρt) vs ρ for various times, and in Figures 3, a typical plot of the mean vs $-\ln[t(1-\rho)^2]/\ln[\rho]$ for some ρ less than 1. From an extensive set of such plots, we can see that $E(Q_{max})$ approaches $-\ln[t(1-\rho)^2]/\ln[\rho]$ as time becomes large. For very small ρ, the deviation from a straight line seems to be periodic in the variables used. Nevertheless, the mean maximum still is monotonic in t. The rapidity with which the mean maximum approaches the speculated limit at $\rho > 1$ and large time can be clearly seen in Figure 2. The departure from the small-time approximation is also clear.

Large Time and $\rho=1$

For large n, and $\rho=1$, the root β closest to zero is $\simeq -[\pi/2n]^2$. Consequently, we would expect that $E(Q_{max}(t))$ to be approximately $\frac{1}{2}\pi t^{1/2}$ for large time, which is consistent with the intuitive belief that $E(Q_{max}(t))$ grows as the standard deviation in a diffusion process or as a random walk with reflection. Indeed, computer simulations confirm this supposition: the exponent for time is ½ (within the limits of the calculation), the coefficient multiplying the time is within 10% of $\frac{1}{2}\pi$, and $\sigma_m(t)$ is approximately $E(Q_{max}(t))/2.4$, at least for $1{,}000 < t < 20{,}000$.

4. RESULTS FOR M/D/1

We carried out Monte Carlo simulations of the M/D/1 system, checked them by running a series of M/M/1 simulations with known answers (thus verifying both the computations based on the mathematics and the simulation), and then compared the numerical results for the M/D/1 system to both the numerical and analytic results on the M/M/1 system.

Large time and $\rho < 1$

When ρ varies from zero to 1 (with $\rho = \lambda t_s$, where t_s is the constant service time, and $t = T/t_s$), the value of $E(Q_{max}(t))$ varies from zero to possibly large values, just as for the M/M/1 system. Therefore, it is of interest to see whether we can find a function of ρ for which the expected means of an M/D/1 system and M/M/1 system would be equal. This certainly can be done for some mean properties of Q; for example(1) we have for an M/M/1 system that $W/t_s|_M$ (Mean time in queue/service time) $= \rho/(1-\rho)$, and for a M/D/1 system: $W/t_s|_D$ (Mean time in Queue/ Service Time) $= \tfrac{1}{2}\rho/(1-\rho)$.

The mapping must tranform 1.0 to 1.0 since the mean queues become infinite in both cases. However, the expected values for both the mean queue (suitably defined) and the maximum queue for an M/M/1 system should be a substantial fraction larger than those for a M/D/1 system whith $\rho = 1$. Therefore, the transformation approximation should not be expected to apply at or close to this limit.

As a first guess, we might try the mapping relating $E(Q)|_M$ to $E(Q)|_D$ since the mapping has suitable properties at small t and small ρ. Unfortunately, this mapping doesn't seem to work for large t, in that it overpredicts $E(Q_{max})|_D$ when ρ is not close to 1. But, *mirabile dictu,* equating $W/t_s|_M$ to $W/t_s|_D$ yields a satisfactory result. That is:

$$E(Q_{max}(t,\rho))|_D \simeq E(Q_{max}(t,\rho_1))|_M$$

where the variable ρ_1 is given by $\rho_1 = \rho/(2 - \rho)$.

This fit is illustrated in Figure 4, which shows the "experimentally" derived mean values vs ρ for an M/D/1 system, and the theoretical mean values calculated for M/M/1 systems with ρ, ρ_1, and matched E(Q)'s. The non-dimensionalized time is 5,000 in these results. Also shown are the "experimentally" derived numbers for an M/M/1 system. It can be seen that they match the theoretical results within the accuracy limits expected from the finite number of simulations.

The M/M/1 and the M/D/1 computations agree very well for times larger than 5,000, and for surprisingly large ρ (but not exceeding 1). Although we cannot expect the means of the maximum queues for M/D/1 and M/M/1 systems to be equal, it is interesting to see that the mapped ρ gives a simulation-derived mean max for ρ very close to 1, and that the larger the time, the closer ρ is allowed to move to 1.

Large time and $\rho > 1$

Since $\lim_{t \to \infty} [E(Q)]$ does not exist if $\rho > 1$, the search for a mapping to convert a nonexistent M/M/1 result to a nonexistent M/D/1 result is somewhat useless. However, as can be seen in Figure 4, our previous speculations on large ρ and large t also apply to the M/D/1 system. The fit of $E(Q_{max}(t))|_M$ to the M/M/1 result (and to $(\rho\text{-}1)t$) clearly improves with increasing ρ and increasing t. In fact, for t=1,000 and $\rho = 1.1$, the difference between the predictions and the simulation results is only 10%; the deviation for t = 1,000 and $\rho = 1.2$ is about 2%, which is the same order of magnitude as the uncertainty in the mean from the simulation, based upon 1,000 runs. As time is increased, the relative fit between the approximation $(\rho\text{-}1)t$ (which is less than the simulated) and the "experimental" results improves, approximately as $t^{-(\frac{1}{2} \text{ to } 1)}$.

Small Time

For very small ρ and only a small number of expected arrivals, $E(Q_{max}(t))$ is about ρt for both an M/M/1 and an M/D/1 system. Even at somewhat larger times, when this approximation is no longer valid, $E(Q_{max}(t))$'s for both systems are still about the same. Only when the expected number of arrivals is more than approximately $O[0.7t^{2/3}]$ does the mean maximum for an M/D/1 system start deviating from the analytical solution for an M/M/1 system. This is illustrated in Figure 4. The figure also shows that the simulated mean max values approach the limiting curve when the expected number of arrivals is approximately $O[1.8t^{2/3}]$.

As can be appreciated from Figure 4, these estimates are rather subjective. Just as we discussed above, for ρ close to 1 there are marked deviations between $E(Q_{max}(t))$ calculated for an M/M/1 system from the governing equations and $E(Q_{max}(t))$ from the simulation of the M/D/1 system. As ρ gets large, this difference disappears.

Large time and $\rho = 1$

For $\rho = 1$, the computational results show that $E(Q_{max}) \simeq 5t^{\frac{1}{2}}/4$ is a good approximation to $E(Q_{max})$ for non-dimensionalized times as low as 25. For times somewhat larger than 25, $\sigma_m(t) \simeq t^{\frac{1}{2}}/6$ is a reasonable fit to the data from the simulation. Thus, both $E(Q_{max})|_D$ and $\sigma_m|_D$ are lower by a factor of approximately $2^{\frac{1}{2}}$ than the corresponding values for an M/M/1 system.

For values of ρ close to 1, we observe a transition for the M/D/1 system, similar to the results on the M/M/1 system, between the $t^{\frac{1}{2}}$ region and the very large time dependency (proportional to ln[t] or t, depending on the sign of ρ-1).

5. CONCLUSIONS

We have shown that $Q_{max}(t)$, the maximum queue encountered from the start to time t, has some interesting properties. For ρ (the utilization factor) less than 1, the mean value of $Q_{max}(t)$ grows as $-\ln[t(1-\rho)^2]/\ln[\rho]$ for large time. For $\rho < 0.2$ (approximately), the variance oscillates periodically in ln[t]; the smaller the ρ, the more pronounced the oscillation. For $\rho < 1$, as time progresses the probability distributions for the maximum queue become only a function of the deviation from the mean and ρ. If $\rho = 1$, then the mean and the standard deviation of $Q_{max}(t)$ vary as $t^{1/2}$, similar to the behavior of a diffusion process. For ρ considerably larger than one, there are simple approximations for the mean, variance, and probability distribution of $Q_{max}(t)$.

If the number of event (arrivals plus departures) is used as the independent variable instead of time, similar results are obtained. Even though the bulk of these results come from a study of M/M/1 systems, many of the results apply to M/D/1, or even G/G/1 systems. A relatively simple transformation seems to closely convert the M/M/1 results to the simulated M/D/1 results for a broad range of times and ρ. Both analytic and simulation techniques were used in this work.

Notation

A/B/m (G/G/l, M/M/1, M/D/1): **m**-server queue where **A** describes the probability distribution for times between arrivals and **B** describes the distribution of the service times, if busy. Specifically, M denotes a Poisson process ("Markov"), D denotes fixed times ("determined"), and G denotes general. It is assumed that the population size and the storage capacity are unlimited.

$A_n(x)$: Determinant of matrix derived from Laplace transform of basic equations forming z_n'. It is an n^{th} order polynomial in x and is implicitely a function of ρ.

β_i: Exponent, function of n, such that $e^{\beta_i t}$ is a solution for the differential equations with state n+1 an absorption state. They satisfy, and in fact are determined by, $A_{n+1}(\beta_i) = 0$.

c_i: Coefficients such that $1 - \Sigma^i c_i e^{\beta_i t} = z_{n+1}$ satisfies the initial conditions $z(0) = z'(0) = z''(0) \ldots = z^{(n)}(0) = 0$.

E(z): Expectation of z, usually in the ensemble sense.

$\Phi_n(\zeta)$: $\rho^{-½n} A_n(\zeta)$. $\Phi_n(\zeta)$ are Tchebychef polynomials.

θ: Variable for Tchebychef polynomials in the form $\sin[(n+1)\theta]/\sin[\theta]$

i, k, n: integers, indices

L{x}: Laplace transform of some function f

λ: Arrival rate, Poisson, T^{-1}

μ: Service rate, Poisson, T^{-1}

Δn: $n - E(Q_{max})$; not necessarily an integer.

ρ_1: $\rho/(2 - \rho)$, such that $W(\rho_1)/t_s$ for an M/M/1 system is $W(\rho)/t_s$ for an M/D/1 system.

σ_a^2: Variance of the arrival intervals; dimension Time^2
(Note: The standard deviation is $\sigma\_$ for this and all other variances.)

σ_m^2: Variance of $Q_{max}(t)$, at a specified time.

σ_Q^2: Variance of the number in the queue (at a specified time); $t \to \infty$ if no time is shown, unless obvious from text.

σ_s^2 : Variance of the service time; dimension Time^2

T: Time; dimensional.

ΔT: Time increment; dimensional.

t: Time, non-dimensionalized, T/t_s.

t_a: Arrival intervals, mean; dimension Time.

$t_{max}(n)$: Time to (first) reach state n (starting in state 0); non-dimensional.

t_s: Service time, mean; dimension Time.

$U_n(\zeta)$: Tchebychef polynomials of the second kind.

W/t_s: Mean time in queue/service time.

x: Variable

$z_i(t)$: Probability that maximum queue encountered, from the start at $t=0$ up to time t, is i or more ($i= 0,1,2\ldots$). It is a function of time. The queue is empty at $t=0$.

O[ ]: Order of magnitude of [ ]

$P_{i,n}$: Probability that system , with an absorption state n+1, is in state i. It is a function of time.

Q: Number in the queue, including any which are being serviced, at time t; if no time is shown, at t→ ∞, unless obvious from text.

$Q_{max}(t)$: Maximum queue encountered from time = 0 to time t. Q_{max} is actually a function of t, ρ, and the shape (i.e., higher moments) of the distribution functions for the arrival and the service times.

ρ: Ratio of the mean time to service to the mean time between arrivals. It is assumed that $\rho > 0$. It is also called the Utilization Factor[(1)].

ζ: Variable, $= \frac{1}{2}(x + \rho + 1)/\rho^{\frac{1}{2}}$.

$[[\beta_1, \beta_2, \ldots, \beta_n]]$: Vandermonde[(7)] matrix used in equation for determining c_i

$$\begin{bmatrix} 1, \beta_1, \beta_1^2, \ldots \beta_1^{n-1} \\ 1, \beta_2, \beta_2^2, \ldots \beta_2^{n-1} \\ 1, \ ., \ ., \ ., \ ., \ . \\ . \\ 1, \beta_n, \beta_n^2, \ldots \beta_n^{n-1} \end{bmatrix}$$

Σ: Sum over index k from 1 to ∞.

Σ^i: Sum over index i for all roots β_i.

{....}: Vector, used for matrix equation for c_i.

′: Differentiation with respect to t.

$\big|_D$: For an M/D/1 system.

$\big|_M$: For an M/M/1 system.

Fig. 1

SIGMA MAX QUEUE vs. TIME
M/M/1 SYSTEM

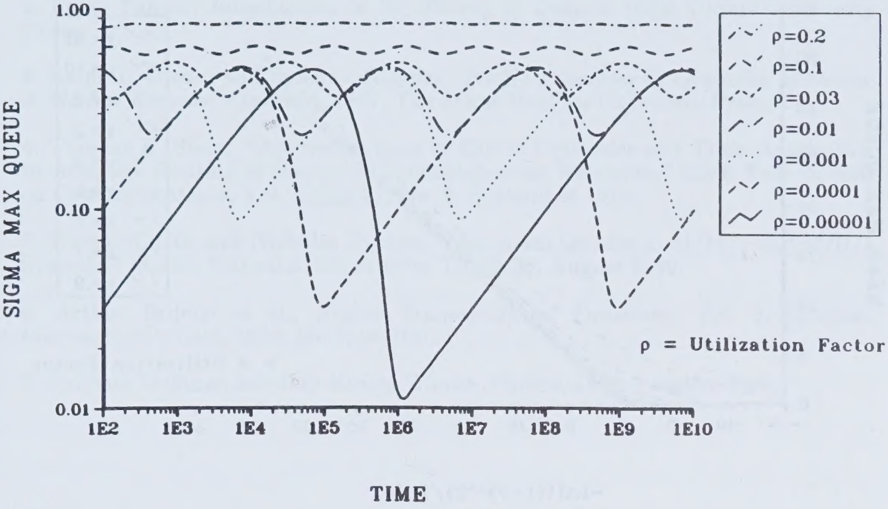


Fig. 2

MEAN MAXIMUM QUEUE / ρt vs ρ, t
M/M/1 SYSTEM

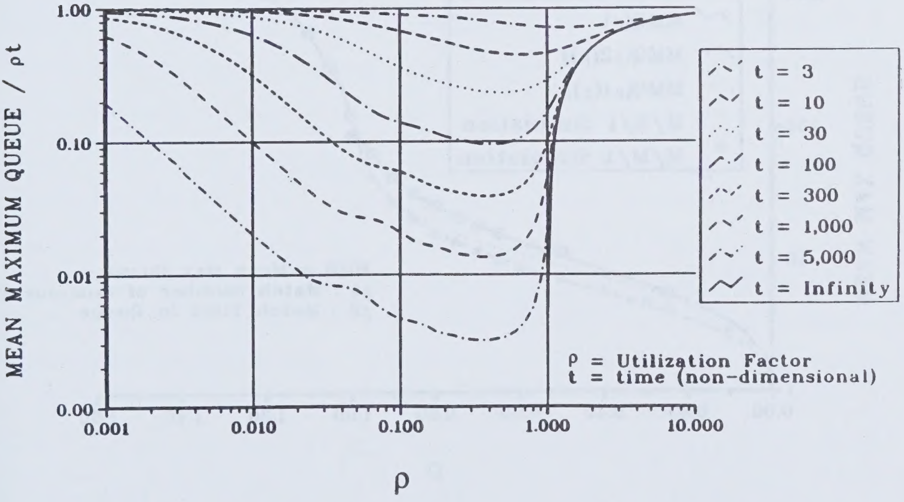


Fig. 3

MEAN MAX QUEUE as F(Time,r)
M/M/1 SYSTEM

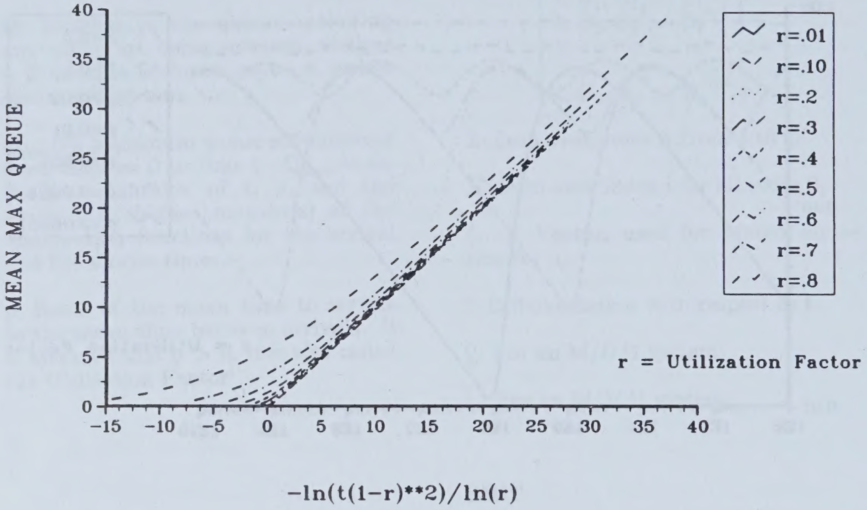
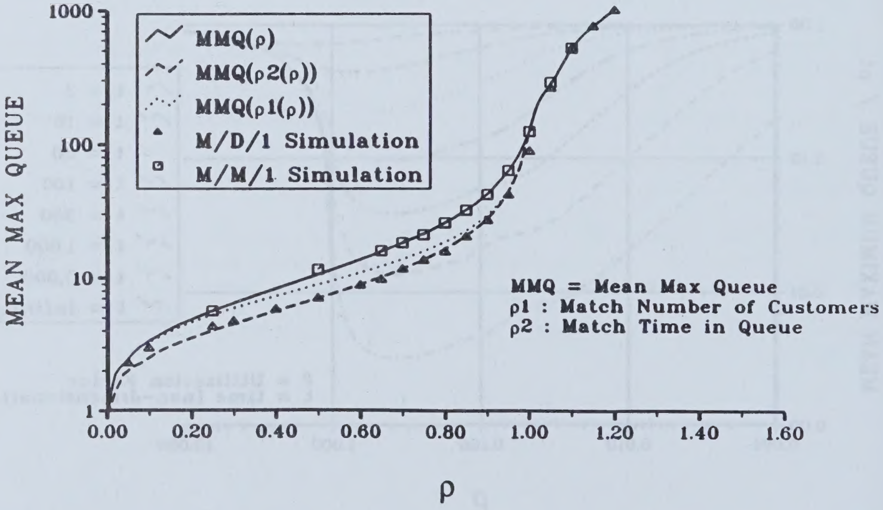


Fig. 4

Comparison: M/M/1 Calculation, Simulation
Time = 5000



REFERENCES

1. Leonard Kleinrock, *Queueing Systems, Volume I: Theory,* 1975, John Wiley & Sons.

2. Lajos Takacs, *Introduction to the Theory of Queues,* 1962, Oxford University Press.

3. Ralph L. Disney and Peter C. Kiessler, *Traffic Processes in Queueing Networks, A Markov Renewal Approach,* 1987, The Johns Hopkins University Press.

4. Thomas E. Stern, "Approximations of Queue Dynamics and Their Application to Adaptive Routing in Computer Communication Networks," IEEE Transactions on Communications, Vol. COM-27, No. 9, September 1979.

5. Werner Furth and Nicholas Duchon "Maximum Queues in M/M/1 and M/D/1 Systems," Martin Marietta Laboratories TR 89-86, August 1989.

6. Arthur Erdelyi et al., *Higher Transcendental Functions,* Vol. 2, Bateman Manuscript Project, 1953, McGraw-Hill.

7. Kenneth Hoffman and Ray Kunze, *Linear Algebra,* 1961, Prentice-Hall.

RELEGATION FOR DECENTRALIZED CONTROL OF COUPLED SYSTEMS[1]

DAVID A. SCHOENWALD and ÜMIT ÖZGÜNER
Department of Electrical Engineering
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210

ABSTRACT

This paper considers the problem of assigning output set-points for optimal decentralized regulation of coupled linear subsystems. The choice of output set-points is constrained by a nonlinear function which is iteratively expanded as a second order power series about an estimate of the current set-point while minimizing the overall cost criterion of the linear quadratic regulator. The basic idea is that the nonlinear constraint function may have an infinite number of feasible output set-point solutions but only one such choice minimizes the overall cost criterion. The decentralized control utilized here consists of proportional state feedback and integral output error feedback for each input/output channel. The feedback gain matrix is obtained via the solution of two coupled Lyapunov equations. The optimal cost will then contain a term which is quadratic in the output set-points, and it is the minimization of this term subject to the nonlinear constraint function which leads to the solution of the optimal choice of output set-points.

I. INTRODUCTION

An increasingly popular approach to control of large scale systems is that of decentralized control. One problem associated with decentralized control is that of managing various system constraints such as sensitivity, input bandwidth, and robustness requirements. Relegation is concerned with the allocation of control effort and information channels to satisfy such constraints while trying to minimize some overall

---

[1] Supported by the Air Force Office of Scientific Research (AFSC) under Contract F49620-89-C-0046.

cost criterion for the system. This paper addresses output set-point relegation which involves the assignment of output set-points to each input/output channel.

Frequently in problems of robotics and slewing of mirrors, flexible structures, etc., one may accomplish a particular task via several different control strategies. For instance, there are many ways in which the individual links of a multi-arm robot can move to achieve a desired end-effector position. The concept of relegation then is to choose the configuration of each robot arm that achieves the desired end-effector position while minimizing overall energy consumption or some other performance criterion. Another example is a multiple mirror setup for ray path tracking. There may be many ways (possibly infinite) to accomplish the desired tracking, but it makes sense to choose the individual mirror settings so as to minimize the energy the mirror motors need to position the individual mirrors.

This paper looks at a rather generic setting for this problem. A linear system with multiple subsystems and coupling (interaction) between these subsystems is considered with the goal being to minimize a standard quadratic cost criterion. The output set-points for each subsystem are not specified exactly but instead are constrained by some nonlinear function which may have an infinite number of solutions. In this paper, an iterative algorithm is proposed in which the nonlinear constraint function is successively quadraticized (expanded as a second order Taylor series) about a current estimate of the output set-points. This estimate of the output set-point is updated by minimizing the quadratic cost criterion with respect to the set-points. This algorithm is continued until satisfactory convergence is obtained. The quadratic cost minimized here is in fact the optimal cost obtained from the solution of the two coupled Lyapunov equations associated with the decentralized optimal control problem treating the set-points as known. Thus, one can view relegation as a further reduction in the "optimal" cost when one treats the set-points as variables instead of just choosing them according to some *ad hoc* method.

Relegation for uncoupled linear subsystems subject to a linear constraint on the set-points has already been investigated in [1]. The generalization to coupled linear subsystems with nonlinear constraints on the output set-points complicates the solution not only because of the nonlinear constraint but also due to the presence of two coupled Lyapunov equations which need to be solved in an iterative manner to determine the quadratic cost function that is to be minimized with respect to the set-points.

II. DECENTRALIZED OPTIMAL CONTROL

The decentralized control framework studied here is obtained primarily from [2,3]. Consider the set of dynamically coupled linear subsystems,

$$\begin{aligned} \dot{x}_i &= \sum_{j=1}^{N} A_{ij} x_j + B_i u_i \\ y_i &= C_i x_i\,, \quad i = 1, 2, \ldots, N \end{aligned} \tag{1}$$

where N is the number of input/output channels, n_i is the dimension of x_i , m_i is the dimension of u_i , and p_i is the dimension of y_i . We wish to minimize the cost criterion

$$J = \frac{1}{2} \sum_{i=1}^{N} \int_0^\infty (z_i^T Q_i z_i + \dot{u}_i^T R_i \dot{u}_i)\, dt \tag{2}$$

where $z_i(t)$ is the $n_i + p_i$ dimension vector defined as

$$z_i(t) = \begin{bmatrix} \dot{x}_i \\ \Delta y_i \end{bmatrix}, \quad \Delta y_i = y_i - y_i^d\,, \quad z_i(0) = \begin{bmatrix} \dot{x}_i(0) \\ \Delta y_i(0) \end{bmatrix} = z_{i0} \tag{3}$$

with Q_i positive semi-definite and R_i positive definite matrices of appropriate dimensions and y_i^d the desired output of the ith input/output channel.

To be able to find the optimal decentralized control law, the following matrices are defined for the overall system

$$\begin{aligned} A &= [A_{ij}]\,, \quad i, j = 1, 2, \ldots, N \\ B &= \text{Block-diag}[B_1, B_2, \ldots, B_N] \\ C &= \text{Block-diag}[C_1, C_2, \ldots, C_N] \end{aligned} \tag{4}$$

with overall dimensions, $n = \sum_{i=1}^{N} n_i$, $m = \sum_{i=1}^{N} m_i$, and $p = \sum_{i=1}^{N} p_i$. Equation (2) can be rewritten as

$$J = \frac{1}{2} \int_0^\infty (z^T Q z + \dot{u}^T R \dot{u})\, dt \tag{5}$$

where Q and R are partitioned as

$$Q = \text{Block-diag}[Q_1, Q_2, \ldots, Q_N], \quad R = \text{Block-diag}[R_1, R_2, \ldots, R_N] \tag{6}$$

and $z(t)$ and $\dot{u}$ are the $n+p$ and m dimensional vectors, respectively, consisting of the N components z_i and $\dot{u}_i$, respectively. The linear system can be redefined in terms of $z(t)$ as

$$\dot{z} = \begin{bmatrix} A_{11} & 0 & A_{12} & 0 & \ldots & A_{1N} & 0 \\ C_1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ A_{21} & 0 & A_{22} & 0 & \ldots & A_{2N} & 0 \\ 0 & 0 & C_2 & 0 & \ldots & 0 & 0 \\ & & & \vdots & & & \\ A_{N1} & 0 & A_{N2} & 0 & \ldots & A_{NN} & 0 \\ 0 & 0 & 0 & 0 & \ldots & C_N & 0 \end{bmatrix} z + \begin{bmatrix} B_1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & B_2 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ & & \vdots & \\ 0 & 0 & \ldots & B_N \\ 0 & 0 & \ldots & 0 \end{bmatrix} \dot{u} = \hat{A}z + \hat{B}\dot{u} \tag{7}$$

with decentralized feedback

$$\begin{aligned} \dot{u}_i &= K_i z_i = K_i^1 \dot{x}_i + K_i^2 \triangle y_i, \quad K_i = [K_i^1 \;\; K_i^2] \\ \dot{u} &= Kz, \quad K = \text{Block-diag}[K_1, K_2, \ldots, K_N] \end{aligned} \tag{8}$$

such that each channel input has the form

$$u_i = K_i^1 x_i + K_i^2 \int_0^t (y_i(\tau) - y_i^d)\, d\tau, \quad i = 1, 2, \ldots, N. \tag{9}$$

The optimal solution for the decentralized feedback gain matrix K to the minimization of the quadratic cost criterion J is known to involve the iterative solution of two coupled Lyapunov equations[2,3]. In particular, the Davidon-Fletcher-Powell method[5] can be used to solve for K via the following algorithm[4]:

1.) Choose $K = K^{(0)}$, $k = 0$

2.) Solve the following two Lyapunov equations for F and P

$$(\hat{A} - \hat{B}K^{(k)})^T F + F(\hat{A} - \hat{B}K^{(k)}) + Q + K^{(k)^T} R K^{(k)} = 0 \tag{10}$$

$$(\hat{A} - \hat{B}K^{(k)})P + P(\hat{A} - \hat{B}K^{(k)})^T + z_0 z_0^T = 0 \tag{11}$$

3.) Set the gradient of J with respect to $K^{(k)}$ to be

$$\nabla_{K^{(k)}} J = (RK^{(k)} - \hat{B}^T F)P \tag{12}$$

4.) Update $K^{(k)}$ using $D^{(k)}$ as the search direction for the kth iteration and ϵ as the update gain

$$K^{(k+1)} = K^{(k)} + \epsilon D^{(k)} \tag{13}$$

5.) Set $k = k + 1$. If $\| \nabla_{K^{(k)}} J \| \leq \delta$, where δ is the tolerance, then stop; otherwise go to step 2.

The optimal cost J^* is then

$$J^* = z_0^T F z_0 \,, \quad F = \begin{bmatrix} F_{11} & F_{12} \\ F_{12}^T & F_{22} \end{bmatrix} . \tag{14}$$

Since it can be assumed that the expected value of $\dot{x}(0) = 0$ and one has no control over terms not containing the output set-points y^d, then the following quadratic cost can be considered

$$J_1^* = y^{d^T} F_{22} y^d - 2y(0)^T F_{22} y^d \tag{15}$$

The problem then is to further minimize J_1^* with respect to y^d subject to a nonlinear constraint function. This problem is addressed in the next section. It should be noted that the above algorithm cannot be solved unless the desired output y^d is known *a priori* since Equation (11) contains the term $z_0 z_0^T$ which depends on y^d. Therefore, the above algorithm will have to be embedded in the algorithm which solves for y^d which is presented in the next section.

III. OUTPUT SET-POINT RELEGATION

At this point we assume there exists a *relegator* whose function is to specify the y^d set-points subject to the following set of nonlinear constraints

$$H_i(y^d) = 0 \,, \quad i = 1, 2, \ldots, N \tag{16}$$

where $H_i(y^d)$ is a scalar function of the output set-points vector y^d. Because general nonlinear functions are difficult to solve, we shall expand $H_i(y^d)$ about a point $\bar{y}$ that represents the current estimate of y^d. The power series is carried to quadratic terms in y^d as follows

$$H_i(y^d) = H_i(\bar{y}) + \frac{\partial H_i(\bar{y})}{\partial y^d}(y^d - \bar{y}) + \frac{1}{2}(y^d - \bar{y})^T \frac{\partial^2 H_i(\bar{y})}{\partial (y^d)^2}(y^d - \bar{y}) = 0, \quad i = 1, 2, \ldots, N \tag{17}$$

where the gradients are defined as

$$\frac{\partial H_i(\bar{y})}{\partial y^d} = \left[\begin{array}{cccc} \frac{\partial H_i(\bar{y})}{\partial y_1^d} & \frac{\partial H_i(\bar{y})}{\partial y_2^d} & \cdots & \frac{\partial H_i(\bar{y})}{\partial y_N^d} \end{array} \right], \quad i = 1, 2, \ldots, N \tag{18}$$

and the Hessians are defined as

$$\frac{\partial^2 H_i(\bar{y})}{\partial (y^d)^2} = \begin{bmatrix} \frac{\partial^2 H_i(\bar{y})}{\partial (y_1^d)^2} & \cdots & \frac{\partial^2 H_i(\bar{y})}{\partial y_N^d y_1^d} \\ \vdots & & \vdots \\ \frac{\partial^2 H_i(\bar{y})}{\partial y_1^d y_N^d} & \cdots & \frac{\partial^2 H_i(\bar{y})}{\partial (y_N^d)^2} \end{bmatrix}. \tag{19}$$

The problem of minimizing J_1^* subject to the constraints (17) is a common optimization problem which can be solved using Lagrange multipliers[6]. Thus, we wish to minimize the following cost criterion with respect to y^d

$$\hat{J} = y^{d^T} F_{22} y^d - 2y(0)^T F_{22} y^d + \sum_{i=1}^{N} \Lambda_i [H_i(\bar{y}) + \frac{\partial H_i(\bar{y})}{\partial y^d}(y^d - \bar{y}) + \frac{1}{2}(y^d - \bar{y})^T \frac{\partial^2 H_i(\bar{y})}{\partial (y^d)^2}(y^d - \bar{y})]. \tag{20}$$

We proceed by setting the gradient of $\hat{J}$ equal to zero and solving for y^d thus obtaining

$$y^d = [2F_{22} + \sum_{i=1}^{N} \frac{\partial^2 H_i(\bar{y})}{\partial (y^d)^2} \Lambda_i]^{-1} [2F_{22} y(0) - \sum_{i=1}^{N} (\frac{\partial H_i(\bar{y})^T}{\partial y^d} - \frac{\partial^2 H_i(\bar{y})}{\partial (y^d)^2} \bar{y}) \Lambda_i] \tag{21}$$

where Λ_i, $i = 1, 2, \ldots, N$, are the Lagrange multipliers and can be solved for by plugging (21) into (17). This motivates the following algorithm to solve for y^d:

1.) Choose $\bar{y} = 0$ initially or some other initial value based on knowledge of the problem.

2.) Evaluate $H_i(\bar{y})$, $\frac{\partial H_i(\bar{y})}{\partial y^d}$, and $\frac{\partial^2 H_i(\bar{y})}{\partial (y^d)^2}$ for $i = 1, 2, \ldots, N$.

3.) Using $\bar{y}$, compute $z_0 z_0^T$ and solve algorithm from Section II for feedback gain K.

4.) Solve for Λ_i, $i = 1, 2, \ldots, N$ by plugging (21) into (17).

5.) Using above, solve for y^d by solving (21) for its p components.

6.) If $\| y^d - \bar{y} \| \leq \delta$ then stop; otherwise set $\bar{y} = y^d$ and go to step 2.

It should be noted that with some sacrifice in accuracy one can linearize (17) instead of expanding to second order and obtain the following simpler equations for Λ_i and y^d

$$\frac{\partial H_i(\bar{y})}{\partial y^d} F_{22}^{-1} \sum_{i=1}^{N} \frac{\partial H_i(\bar{y})^T}{\partial y^d} \Lambda_i = 2[H_i(\bar{y}) + \frac{\partial H_i(\bar{y})}{\partial y^d}(y(0) - \bar{y})] \,, \quad i = 1, 2, \ldots, N \qquad (22)$$

$$y^d = y(0) - 0.5 F_{22}^{-1} \sum_{i=1}^{N} \frac{\partial H_i(\bar{y})^T}{\partial y^d} \Lambda_i \,. \qquad (23)$$

The algorithm above can still be followed to find y^d except Hessian information is no longer needed.

IV. CONCLUSIONS

In this paper, we have presented a procedure which can be followed to determine the optimal set of y^d to minimize a standard quadratic cost criterion with the output set-points constrained by a set of nonlinear equations. This procedure was applied to the proportional-integral regulator problem for decentralized control of coupled linear subsystems. The basic concept is that once an optimal solution has been obtained for this regulator problem, the cost criterion can be further reduced by minimizing the cost with respect to the output set-points provided there is some freedom in choosing these set-points. It is our contention that a great many problems in robotics, flexible structures, and other areas do not require exactly specified output set-points for the individual controllers but are simply constrained by possibly nonlinear functions as determined by the control objectives.

Future work will involve testing of this algorithm on actual examples in robotics and flexible structures. In addition, the theoretical work presented here can be extended to time-varying set-points and ultimately nonlinear interconnected subsystems. Finally, the issue of relegation can be generalized to include state space relegation to input/output channels in an overlapping decompositions framework and to the relegation of feedback resources as needed among the different input/output channels.

V. REFERENCES

[1] Özgüner, Ü., "Decentralized/relegated control," *Proceedings of the 1988 American Control Conference*, Atlanta, GA, 1988.

[2] Geromel, J. C., and Bernussou, J., "An algorithm for optimal decentralized regulation of linear quadratic interconnected systems," *Automatica*, vol. 15, pp. 489-491, 1979.

[3] Khorrami, F., Tien, S., and Özgüner, Ü., "DOLORES: A software package for analysis and design of optimal decentralized control," *Proceedings of the 40th IEEE National Aerospace & Electronics Conference*, Dayton, OH, 1988.

[4] Levine, W. S., Johnson, T. L., and Athans, M., "Optimal limited state variable feedback controllers for linear systems," *IEEE Transactions on Automatic Control*, vol. AC-16, no. 6, pp. 785-793, 1971.

[5] Fletcher, R., and Powell, M. J. D., "A rapidly convergent descent method for minimization," *Computer Journal*, vol. 6, pp. 163-168, 1963.

[6] Luenberger, D. G., *Linear and Nonlinear Programming*, Reading, MA: Addison-Wesley, 1984.

ROBUST CONTROL SYSTEM DESIGN BASED ON PROBABILISTIC DESCRIPTIONS OF PLANT UNCERTAINTY

CHARLES D. SCHAPER, DUNCAN A. MELLICHAMP, AND DALE E. SEBORG
Department of Chemical and Nuclear Engineering
University of California, Santa Barbara, CA 93106

ABSTRACT

In this summary, probabilistic descriptions of plant uncertainty are used for the analysis and design of robust control systems. A measure-theoretic framework is developed to characterize these descriptions in an axiomatic mathematical form. A new robust control system design strategy is developed using the probabilistic uncertainty descriptions. New performance and sensitivity measures are proposed and the Pareto-optimal solutions under a statistical stability constraint are determined by vector optimal programming techniques. The conceptual advantage of this new probabilistic approach over conventional minimax approaches is that the control system is designed for plant descriptions that are probable of representing the true system behavior whereas the minimax control system is designed for a "worst-case" model that can have a small likelihood of describing the actual system behavior.

I. INTRODUCTION

The design of control systems that are robust to model uncertainty is a very active research area. Generally, the current design approaches (e.g. $\mathcal{H}_\infty$ and structured singular value) have used bounds about a nominal model to describe plant uncertainty and then have employed a minimax or "worst-case" design criterion to synthesize a feedback controller (for example, see [1]). Although these approaches have resulted in significant advancement of mathematical control theory, the inherently over-conservative nature of the resulting optimal controller still remains an important issue for practical applications.

Two major problems that induce such conservatism are inherent in the current formulation of the robust control system design problem as a minimax optimization in the presence of bounded uncertainties. First, characterizing process uncertainty by bounds is equivalent to assuming a uniform probability distribution (that is, equal likelihood) of system behavior. This approach neglects the more detailed probabilistic information about the system behavior such as probability distributions for the frequency response [2,3]. Because much of this information is neglected in current design strategies, conservatism is induced into the compensator at the very onset of the design process.

Secondly, although the minimax or "worst-case" design strategy employed by current robust design strategies is important for engineering analysis, it may not be appropriate for optimal synthesis when probabilistic plant information is available. In synthesizing a control system by combining bounded uncertainty descriptions with a minimax criterion, one tends to obtain a feedback compensator that is optimal for a plant description that in reality may have a small likelihood of occurring. Generally, such a compensator is then very conservative at the nominal plant conditions.

To reduce the degree of conservatism, these existing approaches have embedded numerous weighting functions in the control system design strategy. However, there seems to be no clear guideline on how to specify these functions.

In this paper, probabilistic descriptions of plant uncertainty are employed for the purpose of obtaining a less conservative control system design. A new design approach is developed to exploit the probabilistic information by introducing new measures of performance and sensitivity. These measures form a multi-objective criterion whose Pareto-optimal solutions are obtained through vector optimal programming methods. A final control system design can then be obtained by analyzing the performance-robustness trade-offs through CAD techniques.

An overview of the paper is as follows. In Section II, a measure-theoretic analysis of model uncertainty is developed to establish a new mathematical framework. In Section III, the conditions for feedback stability to a statistical level α are developed by using the concept of entropy to efficiently structure the uncertainty regions. A new design approach is presented in Section IV with a numerical example provided. Due to space limitations, proofs and a number of details of the developments have been omitted. These can be obtained from the authors.

II. MEASURE-THEORETIC CHARACTERIZATION OF UNCERTAINTY

In this section, a new framework is summarized to characterize model uncertainty descriptions in a fundamental mathematical form. This framework is completely described by the authors in [4,5]. In addition, the "deterministic" approach of describing plant uncertainty by bounds is encompassed by this framework.

A measure-theoretic approach is used to characterize model uncertainty in the sense of Kolmogorov. In developing this mathematically detailed formulation, both time-domain and frequency-domain model uncertainty descriptions are now mathematically well-defined, hence fundamental properties such as the regularity property of separability, in addition to existence, continuity, and differentiability properties, can be inferred. Moreover, this framework provides the mathematical basis to perform operations, such as expectations, on probabilistic descriptions of plant uncertainty.

Before summarizing the measure-theoretic formulation of plant uncertainty, the system signals are characterized in a stochastic representation. Discrete time systems are considered. The representation described by Caines [6] is employed here. The signals (inputs, outputs, and disturbances) of the system are represented as stochastic processes associated with the probability space $(\Omega, \mathcal{F}, Q)$ where $\Omega = R^T$ where R is the real number field and T denotes the discrete time index set, $\mathcal{F} = \mathcal{B}(R)^T$ is the σ-algebra of Borel sets on the real number field R and Q is the probability measure on $\mathcal{B}(R)^T$. These random variables are characterized in a Hilbert space $L^2(\Omega, \mathcal{F}, Q)$ setting. The input-output mapping of the open-loop plant is presently represented by the quadropole $((\Omega, \mathcal{F}, Q), \mathcal{U}, \mathcal{Y}, \zeta)$ where $\mathcal{U}$ and $\mathcal{Y}$ denote the set of input and output sequences respectively, and ζ represents a linear, time-invariant mapping between the input and output.

The proposed measure-theoretic approach of describing plant uncertainty is now developed. See Billingsley [7] for the mathematical background of measure theory. Consider the frequency domain function space of all time-invariant mappings. We characterize this function space as a probability measure space with the product space C^W where C is the complex number field and $W = [-\pi,\pi]$ is the indexing interval of frequencies. The measure space is completely specified by the probability measure P on the σ-algebra of Borel sets $\mathcal{B}(C)^W$. A path function or realization on this probability space is denoted by $x(j\omega,\xi)$ which takes on complex values at each frequency ω. For every frequency-domain path function in the measure space there corresponds a time-domain linear, time-invariant mapping function ζ.

The mathematical characteristics of the path functions of the measure space $(C^W, \mathcal{B}(C)^W, P)$ are now analyzed in the measure-theoretic sense of Kolmogorov. First, consider the real rational transfer function defined on the unit circle $\{z = \exp(j\omega) : -\pi \leq \omega < \pi\}$ with a sampling period of one

$$G(e^{j\omega}, \theta) = K \frac{\prod_{i=1}^{b} (1 - \beta_i z^{-1})}{\prod_{i=1}^{a} (1 - \alpha_i z^{-1})} . \tag{1}$$

Define $\theta = (K, \alpha_i ,.., \beta_i ,..)$ as a random vector variable comprised of real valued elements. Allow the model orders $a,b \to \infty$ with θ defined on the Cartesian product $R^\infty = \prod_{i=1}^{\infty} R_i$ with Borel σ-algebra $\mathcal{B}(R)^\infty = \prod_{i=1}^{\infty} \mathcal{B}(R)_i$ and a consistent probability measure L on $\mathcal{B}(R)^\infty$. Thus, a realization of $G(e^{j\omega}, \theta)$ is characterized as a linear, time-invariant function with coefficients drawn from a given distribution.

Consider the map $G(e^{j\omega}, \theta)$: $R^\infty \to C^\infty$ where $C^\infty = \prod_{i=1}^{\infty} C_i$.

Theorem 1. $G(e^{j\omega}, \theta)$ is measurable (a random process), stated in mathematical form as

$$[G(e^{j\omega}, \theta)]^{-1} (D_\omega) = \{\psi : G(e^{j\omega}, \psi) \in D_\omega\} \in \mathcal{B}(R)^\infty \ \forall \ D_\omega \in \mathcal{B}(C)^\infty \tag{2}$$

provided that the measurable space $(C^\infty, \mathcal{B}(C)^\infty)$ is $C^\infty = \prod_{i=1}^{\infty} C_i$ and $\mathcal{B}(C)^\infty = \prod_{i=1}^{\infty} \mathcal{B}(C)_i$. That is, the product space is taken over a countable set of frequencies. This countable set is taken such that the inverse image of $G(e^{j\omega}, \theta)$ for $\mathcal{B}(C)^\infty$ is contained on $\mathcal{B}(R)^\infty$. ∇∇

The derived distribution for the random process $G(e^{j\omega}, \theta)$ is denoted as J. Thus, the parameter coefficients defined on $(R^\infty, \mathcal{B}(R)^\infty, L)$ induces through the random process $G(e^{j\omega}, \theta)$ the probability space $(C^\infty, \mathcal{B}(C)^\infty, J)$ defined at each frequency on the complex plane.

This finite dimensional distribution J can be related to the measure space generated by the space of all complex-valued path functions $x(j\omega,\xi)$ by applying some basic theorems of measure theory to our problem.

Theorem 2. If the probability measure space (C^w, $\mathcal{B}(C)^w$, P) consists of continuous functions with probability 1, then the separable process represented by (C^∞, $\mathcal{B}(C)^\infty$, J) also consists of continuous functions with probability 1. ∇∇

Consequently, there exists an equivalency between the space of real rational transfer functions and frequency domain uncertainty regions. The conditions to generate P such that (C^w, $\mathcal{B}(C)^w$, P) consists of continuous functions with probability 1 is given in [4].

This mathematical framework can be extended to nonlinear or time-varying systems [4]. For example, suppose frequency domain bounds have been specified to account for time variations in the process model. The equivalent random process corresponding to a time-varying parametric model can be obtained as follows. Consider the not necessarily infinite dimensional linear model with time varying coefficients λ defined by the joint probability distribution function $\rho_1(\lambda(\cdot); t, t+1, ..)$. Although the random vector λ may be nonstationary over a finite period of time, it can be taken as a stationary process as $t \to \infty$ with a time averaged distribution function $\bar{\rho}_1(\lambda)$ on the measurable space (R^d, $\mathcal{B}(R)^d$) where $d = \dim(\lambda)$. Then, the frequency-domain probability measure of the uncertainty region will extend to the measure $\bar{\rho}_1(\lambda)$.

III. α-STABILITY, PERFORMANCE AND RISK

Consider the feedback control system shown in Figure 1. In our framework, the frequency response of the plant model, $G(z^{-1})$, is represented as a complex-valued random variable associated with the probability measure space (C^w, $\mathcal{B}(C)^w$, P). The compensator is denoted by $C(z^{-1})$ and the system signals are characterized as stochastic processes in (Ω, $\mathcal{F}$, Q). The internal and external stability of the feedback system can be determined through the transfer matrix [8]

$$H(z^{-1}) = \frac{1}{1 + G(z^{-1})C(z^{-1})} \begin{bmatrix} 1 & -G(z^{-1}) \\ C(z^{-1}) & 1 \end{bmatrix} \tag{3}$$

The pair $[G(z^{-1}), C(z^{-1})]$ is stable if and only if $\mathbf{H}(z^{-1}) \in \mathbf{RH}_\infty \; \forall \; G(z^{-1})$ whose frequency response is contained within the uncertainty region. However, roughly speaking, this spectral uncertainty region may contain plant descriptions that have a "small" likelihood of occurring. In order to reduce the conservatism of the control law, this subset of plant descriptions should not be considered in the stability analysis. Only the set of plants which cover the uncertainty region to a desired degree of statistical accuracy should be considered. In the following development, the necessary condition for stability to a statistical level α for frequency-domain probabilistic descriptions of plant uncertainty is developed. Stengel and Ryan [9] have analyzed the statistical level of stability for a parametric characteristic equation comprised of stochastic coefficients.

In order to provide a fundamental mathematical basis to structure probabilistic uncertainty descriptions for stability analysis, the mathematical concept of entropy is employed. This formulation can be used to quantify the information content of a general bounded uncertainty region. Presently, the objective is to determine the maximum entropy bounds on a spectral uncertainty region at each frequency that cover the region to a statistical level α, $0 \leq \alpha \leq 1$.

Consider the partitioning or ε-covering of C as the collection of hollow spheres, $S_o(\varepsilon)$, of the type $\{ G \in C;\ \| G - \tilde{G} \| - \| G_o - \tilde{G} \| \leq \varepsilon \}$ $\forall\ G_o \in C$ with center $\tilde{G}$, $\varepsilon > 0$, whose union is equal to the complex number field C. Let $n(\varepsilon)$ denote the infinum of the number of hollow spheres in an ε-covering of C taken over all possible coverings. The entropy of C, denoted by $I_X(\varepsilon)$, is defined as

$$I_X(\varepsilon) \equiv - \sum_{n(\varepsilon)} P(G_o,\varepsilon) \log P(G_o,\varepsilon) \ \leq\ \log n(\varepsilon) \tag{4}$$

where $P(G_o,\varepsilon)$ is the probability mass of the hollow sphere at G_o with radial distance ε. The quantity $I_X(\varepsilon)$, can be considered to be a measure of the uncertainty contained within C at each frequency. In the following development, the limit of $I_X(\varepsilon)$ as $\varepsilon \rightarrow 0$ is employed. To obtain a subset $B_\alpha(\varepsilon) \subset C^W$ of plants that maximize the entropy function $I_X(\varepsilon)$ at each frequency and cover C^W to an α-level probability at each frequency, the problem is formulated as

$$\max_{t_\omega(\varepsilon) \subset S_o(\varepsilon)} \ - \sum_{t_\omega(\varepsilon)} P(G_o,\varepsilon) \log P(G_o,\varepsilon) \tag{5}$$

$$\text{s. t.} \ \sum_{t_\omega(\varepsilon)} P(G_o,\varepsilon) = \alpha \tag{6}$$

as $\varepsilon \rightarrow 0$. The parameter α is used to indicate a desired statistical level of feedback stability. This analysis is performed $\forall\ \omega \in [0, \infty)$ to obtain $B_\alpha(\varepsilon)$ which is comprised of $\{t_\omega(\varepsilon)\}\ \forall\ \omega$.

This maximum entropy approach can be used to determine the mean and statistical bounds for any given uncertainty structure. Presently, this approach is used to obtain bounds for two probabilistic uncertainty descriptions.

Lemma 1. Consider the disk representation of model uncertainty in which a polar-coordinate normal distribution is assigned as a function of the disk radius. The maximum entropy set $B_\alpha(\varepsilon)$ is given by

$$B_\alpha(\varepsilon) = \{ G(e^{-j\omega}) : | G(e^{-j\omega}) - \hat{G}(e^{-j\omega}) | \leq \beta\Delta_a(\omega) \,;\, T(\beta) = \alpha \} \tag{7}$$

where $\hat{G}(e^{-j\omega})$ is the mean and $\Delta_a(\omega)$ is the variance of the probability distribution for the disk radius as a function of frequency. The transformation $T(\beta)$ gives the proper scaling of β for the given distribution to obtain the α-level ε-covering of C at each frequency. ∇∇

Lemma 2. Consider the case where a uniform probability distribution is assigned to the disk radius of Lemma 2. In this case, there is no unique set $B_\alpha(\varepsilon)$ satisfying (5) and (6) if $\alpha < 1$. Only in the case $\alpha = 1$ will a unique set be obtained. $\nabla\nabla$

These maximum entropy bounds can then be employed in conjunction with the small gain theorem (for example, [1]) to form the necessary condition for stability to a statistical level α $\forall$ ω, henceforth referred to as α-stability. The necessary condition for α-stability is given by

$$\| \tilde{\eta}(e^{-j\omega})\, \Delta_m(e^{-j\omega}) \|_\infty \triangleq \sup_{\omega} | \tilde{\eta}(e^{-j\omega})\, \Delta_m(e^{-j\omega}) | < 1 \tag{8}$$

where the complimentary sensitivity function at nominal conditions $\tilde{\eta}(e^{-j\omega}) = \hat{G}(e^{-j\omega}) C(e^{-j\omega})/(1+\hat{G}(e^{-j\omega})C(e^{-j\omega})$ and $\Delta_m(e^{-j\omega}) = T^{-1}(\alpha)\Delta_a(e^{-j\omega}) \,/\, | \hat{G}(e^{-j\omega}) |$. The inverse transformation $T^{-1}(\alpha)$ is found through (8), for example. The set of compensators which stabilize the plant to the statistical level α is then expressed as

$$C_\alpha = \{ C(e^{-j\omega}) : \mathbf{H}(e^{-j\omega}) \in \mathbf{RH}_\infty\ \forall\ G(e^{-j\omega}) \in B_\alpha(\varepsilon) \} \tag{9}$$

An alternative approach of obtaining an expression for this α-stabilizing set of compensators is given by the authors [5].

A new design approach is developed to exploit the probabilistic description of plant uncertainty. This approach is based upon an explicit separation of performance and sensitivity measures to form a multiple-objective design criterion. The performance and sensitivity measures are developed from the model uncertainty induced first and second order statistical moments of a general (or user-defined) cost function. In the present formulation, the cost function is chosen as the 2-norm of the feedback error signal.

First, consider the spectral density of the feedback error signal which will serve as a performance measure,

$$\Phi_e(z^{-1}) = S(z^{-1})\, \Phi_c(z^{-1})\, S(z^{-1})^* \tag{10}$$

where the sensitivity function $S(z^{-1}) = 1/(1 + G(z^{-1})C(z^{-1}))$ and the spectrum $\Phi_c(z^{-1}) = \Phi_r(z^{-1}) + \Phi_v(z^{-1})$. The spectral density function of the disturbance $v(t)$ is expressed as

$$\Phi_v(z^{-1}) = \sum_{\tau=-\infty}^{\infty} \mathcal{E}_{M|P}\, [v(t)\, v(t-\tau)]\, z^{-\tau} \tag{11}$$

The conditional expectation is taken with respect to the measure $M = Q \times P$ that is conditioned on the measure P. Usually, the random processes associated with the model uncertainty and the disturbances are statistically independent and thus $\mathcal{E}_{M|P} = \mathcal{E}_Q$. The spectral density of the command signal can be formed in a similar fashion and can also be represented in quasi-stationary form [2].

The spectral density $\Phi_e(z^{-1})$ is a random variable over frequency since it is a function of the probabilistic plant $G(z^{-1})$. The expected value of the cost function as defined in (10) is given by

$$\bar{\Phi}_e(z^{-1}) = \int_{\Omega} \Phi_e(z^{-1})\, dP \tag{12}$$

Definition 1. The *probabilistic performance index* is given by the weighted cost function

$$P_B[\, C(z^{-1})\,] = \frac{1}{2\pi j} \oint_{|z|=1} W_1(z^{-1})\, \bar{\Phi}_e(z^{-1}) \frac{dz}{z} \tag{13}$$

The weighting function $W_1(z^{-1})$ is imposed to increase the flexibility of the design approach, however in most cases it should be set to 1.

The performance index expressed by (13) is used to indicate the average fidelity of the feedback control system. The sensitivity of the feedback control system to model uncertainty is also of concern. In the present context, sensitivity is considered as the model-uncertainty induced deviation of the performance index from its expected value. Thus, the probabilistic sensitivity or robustness of the control system is expressed as the expected relative departure from $\bar{\Phi}_e(z^{-1})$ and is given by

$$\bar{\Phi}_\rho(z^{-1}) = \int_{\Omega} [\, \Phi_m(z^{-1})\, \Lambda(z^{-1})\, \Phi_m(z^{-1})^*\,]\, dP \tag{14}$$

where

$$\Phi_m(z^{-1}) = \Phi_e(z^{-1}) - \bar{\Phi}_e(z^{-1})$$

$$\Lambda(z^{-1}) = \bar{\Phi}_e(z^{-1})^* \bar{\Phi}_e(z^{-1})$$

Definition 2. The *probabilistic risk* which serves as the measure of the feedback control system sensitivity, is given by

$$R_B[\, C(z^{-1})\,] = \frac{1}{2\pi j} \oint_{|z|=1} W_2(z^{-1})\, \bar{\Phi}_\rho(z^{-1}) \frac{dz}{z} \tag{15}$$

where $W_2(z^{-1})$ is a weighting polynomial which has been introduced for design flexibility but would normally be set to 1.

Essentially, the probabilistic risk is the second moment of the uncertainty-induced relative discrepancy of the cost function. In general terms for a cost function J, $R_B = \mathcal{E}[(J - \bar{J}) / \bar{J}]^2$ where the expectation is taken with respect to the joint measure of model uncertainty and disturbances. Thus, the sensitivity measure is a function of the entire uncertainty region (as opposed to a single point) but employs the measure P to place more emphasis on plant descriptions that possess a greater likelihood of describing the actual system behavior.

IV. A NEW DESIGN APPROACH

The measures of control system performance (13) and sensitivity risk (15) comprise a vector-valued objective functional $F \equiv (f_1[C(z^{-1})] \; f_2[C(z^{-1})])^T = (P_B[C(z^{-1})] \; R_B[C(z^{-1})])^T$. It is well known that performance and sensitivity constitute conflicting objectives. Thus, the trade-offs of these conflicting objectives can be explicitly analyzed using the theory of vector-optimality. In the following development, the feedback compensators are determined that are Pareto-optimal for the multi-objective criterion F and are robustly stable to a statistical level α. This problem is formulated as determining the Pareto-optimal set of feedback compensators $C_\alpha^{opt} = \{ C_*(z^{-1}) \} \subset C_\alpha$. A compensator belongs to this set if there does not exist $C_o(z^{-1}) \in C_\alpha$ such that

$$\text{i) } f_i[C_o(z^{-1})] \le f_i[C_*(z^{-1})] \text{ for } i = 1,2, \text{ and}$$

$$\text{ii) } f_j[C_o(z^{-1})] < f_j[C_*(z^{-1})] \text{ for some } j = 1,2$$

with the constraint that $C_*(z^{-1}) \in C_\alpha$.

A numerical example is presented to demonstrate the new approach to robust control system design using probabilistic descriptions of plant uncertainty. Vector optimal programming techniques are employed to solve this multiple-objective criterion. This example provides the set of Pareto-optimal controllers C_α^{opt} for a given compensator structure.

Example. Consider the system model

$$y(t) = G(z^{-1})u(t) + H(z^{-1})e(t) \tag{16}$$

where e(t) is a zero mean, white, with variance 1, and

$$\hat{G}(z^{-1}) = \frac{z^{-1}}{1 + 0.7z^{-1}} \; ; \; H(z^{-1}) = \frac{1 + 0.3z^{-1}}{1 + 0.7z^{-1}}$$

The probabilistic uncertainty is given by

$$\text{std dev}\{ \lim_{z \to 1} [G(z^{-1})] \} = 0.2 \tag{17}$$

A controller of the form

$$C(z^{-1}) = \frac{k}{1 + cz^{-1}} \tag{18}$$

is to be designed to minimize an LQG performance index with a weighting of four on the input energy term and to stabilize the system for bounds of three standard deviations on the uncertainty. The values of performance and risk functions corresponding to the set of Pareto-optimal solutions are presented in Figure 2 where $P_B[\ C(z^{-1})\ ]$ and $R_B[\ C(z^{-1})\ ]$ have been normalized by their respective minimum values. The optimal LQG design with the nominal model corresponds roughly to the point (8,1) which is off the figure. This design corresponds to a high risk, high performance design. Therefore, this example also demonstrates the known high sensitivity of the LQG methodology. To obtain a single design from the set of Pareto-optimal solutions C_α^{opt}, one can employ a scalar functional obtained by appropriate weighting of F.

V. CONCLUSIONS

A new approach has been developed that introduces new concepts and theory for robust control system analysis and design. The proposed approach has been based on describing plant uncertainty by probabilistic descriptions. A measure-theoretic framework has been developed to mathematically analyze these descriptions and has been shown to encompass the "deterministic" approach of bounded uncertainty descriptions. Frequency-domain conditions have been developed to ensure robust stability to a statistical level α. A new design strategy has been developed using the probabilistic descriptions of plant uncertainty.

REFERENCES

[1] M. Morari and E. Zafiriou, *Robust Process Control*. Englewood Cliffs: Prentice-Hall, 1989.

[2] L. Ljung, *System Identification: Theory for the User*. Englewood Cliffs, N.J.: Prentice-Hall, 1987.

[3] G. C. Goodwin and M. E. Salgado, "Quantification of uncertainty in estimation using an embedding principle," *Proc. Amer. Control Conf.*, Pittsburgh, pp. 1416-1421, 1989.

[4] C. D. Schaper, "Measure-theoretic formulation of state/spectral model uncertainty descriptions," *research report*, Department of Chemical and Nuclear Engineering, University of California, Santa Barbara, September 1989.

[5] C. D. Schaper, D. A. Mellichamp, and D. E. Seborg, "A new approach for robust control system analysis and design," submitted for the *1990 American Control Conference.*

[6] P. E. Caines, *Linear Stochastic Systems*. New York: John Wiley & Sons, 1988.

[7] P. Billingsley, *Probability and Measure*. New York: John Wiley & Sons, 1986.

[8] M. Vidyasagar, *Control System Synthesis: A Factorization Approach*. Cambridge: MIT Press, 1987.

[9] R. F. Stengel and L. E. Ryan, "Stochastic robustness of linear control systems," *Proc. Conf. on Information Sciences and Systems*, Johns Hopkins Univ., Baltimore, 1989.

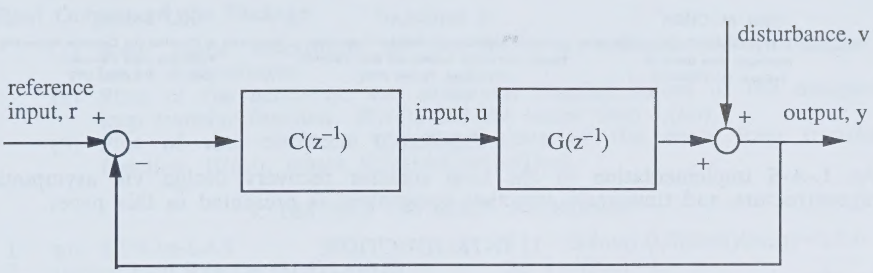

disturbance, v
reference input, r
+
−
input, u
C(z⁻¹)
G(z⁻¹)
+
+
output, y

Figure 1. Feedback control system

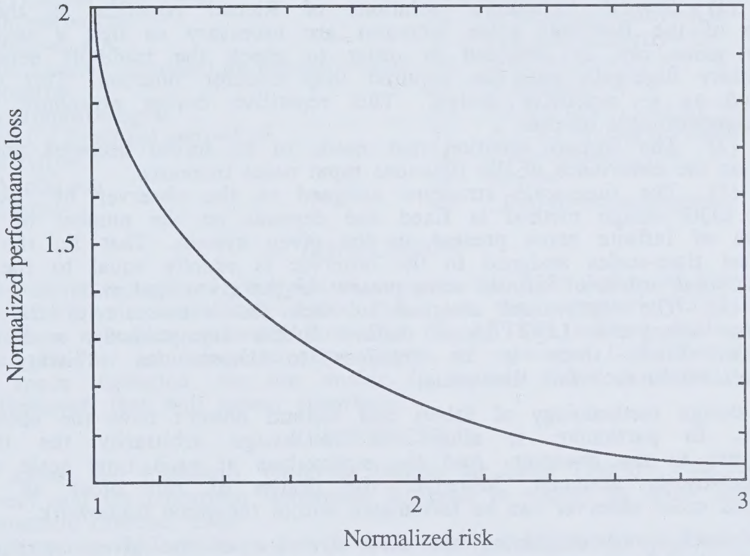


Figure 2. Performance and sensitivity loss of Pareto-optimal feedback compensators

A C-A-D PACKAGE FOR LTR DESIGN VIA ASYMPTOTIC EIGENSTRUCTURE AND TIME-SCALE STRUCTURE ASSIGNMENT

BEN M. CHEN†
Department of Electrical and Computer Engineering
Washington State University
Pullman, WA 99165-2752

S. BINGULAC
Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

ALI SABERI†
Department of Electrical and Computer Engineering
Washington State University
Pullman , WA 99165-2752

ABSTRACT

An L-A-S implementation of the loop transfer recovery design via asymptotic eigenstructure and time-scale structure assignment is presented in this paper.

1. INTRODUCTION

In this paper, we present a software package in L-A-S (Linear Algebra and Systems language),[2] for loop transfer recovery (LTR) design. The approach for solving LTR problem in this package is based on the asymptotic eigenstructure and time-scale structure assignment (AETSSA) design methodology, developed by Saberi and Sannuti (1988),[1]. As it was pointed out in [1], this method has several advantages over the asymptotic LQG design theory, [4,6]. There are a number of difficulties posed by the asymptotic LQG design as well as other design methods for LTR. The restrictions posed by the asymptotic LQG design method can be summarized as:

(1) Several 'repetitive' solutions of Riccati equation as the covariance of the ficticious noise increases are necessary so that a sequence of filter gains can be obtained in order to check the trade-off between the necessary high-gain and the required loop transfer function. This could be viewed as a 'repetitive design'. This repetitive design procedure increases the computational burden.

(2) The Riccati equation that needs to be solved becomes numerically stiff as the covariance of the ficticious input noise increases.

(3) The time-scale structure assigned to the observer by the asymptotic LQG design method is fixed and depends on the number of different orders of infinite zeros present in the given system. That is, the number of fast time-scales assigned to the observer is exactly equal to the number of different orders of infinite zeros present in the given system.

(4) The eigenvalues assigned to each fast time-scale of the observer by the asymptotic LQG design method follow asymptotically a Butterworth pattern. Thus, there is no freedom to choose an arbitrary set of eigenvalues in each fast time-scale.

The design methodology of Saberi and Sannuti doesn't have the above drawbacks. In particular, it allows us to assign arbitrarily the time-scale structure to the observer. And the eigenvalues at each time scale also can be arbitrarily assigned. Moreover the design of full order as well as reduced order observer can be formulated within the same framework.

Our package, by examining the zero structure of the given system, determines whether or not exact LTR and/or almost LTR is achievable. Based on this and the user's choice of the observer (full order or reduced order), then the design can be completed.

2. SOFTWARE PACKAGE DESCRIPTION

Primary Input Data Requirement:

(1) State space matrices (A,B,C) and the regulator gain matrix F (for

---

† The work of B.M. Chen and A. Saberi is supported by NSF under grant Number ECS-8618953.

the case of recovery at the input node).
(2) Time-scale structure and desired eigenvalues corresponded to fast time-scales.

Final Outputs of the Package:

(1) State space description of the (full or reduced order) observer-based compensator.
(2) Plots of the maximum and minimum singular values of the designed loop transfer function, $\mathscr{L}(j\omega)$, and the target loop $L(j\omega)$.
(3) Plot of the maximum singular values of the error loop transfer function, $E(j\omega)$, where $E(j\omega)=L(j\omega)-\mathscr{L}(j\omega)$.

3. LISTING OF MAIN PROGRAM

```
   nli; LTR-in-LAS
   (rbf)=Ascb,Rxa,Qv,G1,G2,[ltrB]
   (rbf)=A,B,C,Lf,Laf,Lab,Lbf,Cs,[ltrA]
   Qv(fsys,sub)=P,mV
   P(-1),Lf(*)=L
   P(rdi),A(rdi)(mcp)=nf,n
   n,nf(-)=r
   r,Rxa(-)=nb
   r,r(dim)=Ir
   Ir,P(dia,sub)=P
   P(-1),Ascb(*),P(*)=Altr
   0,0(dzm)=Kbb
   nb(ifj)=r,r,A
A:Altr,Rxa(inc),Rxa(inc),nb,nb(exm)=Abb
   nb(out)=
   nb,2(inpm)=EgvB
   Abb,Cs,EgvB(ppl,sub)=Kbb
r:doc1(tfi)=
   mV(cdi)=m
   m(out)=
   0(dma),0,0(dzm)(mcp)=i,Ko
I :i(inc)=i
   i,m(ifj)=C,C,D
C:mV,1,i(exm)=qi
   qi(out)=
   (dsc)=sgma
   qi,2(inpm)=egvs
   sgma,qi,egvs(siso,sub)=Ki
   Ko,Ki(dia,sub)=Ko
   (jmp)=I
D:Ko,L(+)=K
   nf,Kbb(cdi)(dzm)=Kfb
   Lab,Kbb,Kfb(rti)=Kb
   Laf,Lbf,K(rti)=K
   K,Kb(cti)=K
   G1,P(*),K(*),G2(-1)(*)=K
   A,K,C(*)(-)(egv)=EgOB
   K,EgOB(out,t,1)=
   (sto)=
   _End-of-main-program.
```

4. CONCLUSION

We have solved a bank of examples to compare the LTR design of Saberi and Sannuti with the conventional method based on the asymptotic LQG design theory. These examples uniformly show that the AETSSA method yields smaller observer gain for achieving the same degree of recovery. Unfortunately, due to the space limitation, we are unable to include these examples in this paper. However, they will appear elsewhere.

REFERENCES

[1] A. Saberi and P.Sannuti, Observer Design for Loop Transfer Function Recovery and for Uncertain Dynamical Systems, to appear in IEEE Trans. on Automatic Control, 1989

[2] S. Bingulac, CAD package L-A-S: A Research and Educational Tool in Systems and Control, Proc. 20th Southeastern Symposium on System Theory, Charlotte, NC, March 20-22, 1988, pp. 44-49.

[3] P. Sannuti and A. Saberi, Special Coordinate Basis for Multivariable Linear Systems-Finite and Infinite Zero Structure, Int. J. Control, Vol 45, No.5, 1987, pp.1655-1704.

[4] J.C. Doyle and G. Stein, Multivariable Feedback Design : Concepts for a Classical/Modern Synthesis, IEEE Trans. on Auto. Control, AC-26, 1981

[5] B.M. Chen, Software Manual for the Special Coordinate Basis for Multivariable Linear Systems, Washington State University Report ECE-094, 1988.

[6] D.B. Ridgely and S.S. Banda, Introduction to Robust Multivariable Control, Report AFWAL-TR-3102, Feb. 1986.

PERFORMANCE AND STABILITY ROBUSTNESS OF LEB (LINEAR-ELLIPSOIDAL-BOUNDED) COMPENSATORS

CLAUDIO D ANTONINI
Residential Address: Buenos Aires Institute of Technology
Electronic Engineering Department
Buenos Aires, Argentina

Abstract

The stability robustness analysis of linear time-invariant systems (LTI) is a worst-case design. Usually we are interested in performance robustness. If the performance robustness goals are satisfied, the stability robustness goals are naturally accomplished. For example, if a stable closed-loop system retains its bounded-input-bounded-output behavior when it is subject to disturbances or modelling errors, it will surely be away from instability.

In this paper the solution to the multiple disturbances case for multiple-input multiple-output (MIMO) LTI plants is presented, when the inputs, the outputs (or the state variables) and the measurement errors are bounded by bounds chosen by the designer. Also, the robustness of the LEB regulator will be discussed. Examples will be given that show a) how the maximum input disturbances are found in the presence of measurement noise while satisfying the control and output bounds (performance robustness), and b) improved stability robustness behavior with respect to the use of linear quadratic (LQ) regulators when applied to the AWJSR Aircraft.

Introduction and review of previous results

In this section the basic results of previous works related to the derivation of the single-disturbance case for the LEB regulator [23,13] will be introduced. We may represent the LTI system by:

$$dx(t)/dt = Ax(t) + Bu(t) + Gw(t) \tag{1}$$

$$y(t) = Cx(t) + Hv(t) \tag{2}$$

where dim(x)=nx1, dim(u)=mx1, dim(w)=ndx1, dim(y)=lx1 and the matrices are of appropriate dimensions. This system will be controlled by a constant gain regulator that generates a control law

$$u(t) = Kx(t) \tag{3}$$

Using (3), (1) and (2) may be rewritten as follows

$$dx(t)/dt = (A + BK)x(t) + Gw(t) \tag{4}$$

$$y(t) = Cx(t) + Hv(t) \tag{5}$$

It was shown in [23] and [20] that if the system described by (4) and (5) is perturbed by a vector of (statistically) unknown-but-bounded disturbances w, and the ellipsoidal set that contains the disturbance w is represented by

$$\Omega_w = \{w : w' \, Q^{-1} \, w \leq 1\} \tag{6}$$

where dim(Q)=nd x nd, the following facts may be observed:

1. the state x of the system (4)-(5), subject to the disturbance (6) is contained within the ellipsoid described by

$$\Omega_x = \{x : x' \, \Gamma^{-1} \, x \leq 1\} \tag{7}$$

$$d\Gamma(t)/dt = (A+BK)'\Gamma + \Gamma(A+BK) + \beta(t)\Gamma + \frac{G'QG}{\beta(t)} \tag{8}$$

and $\Gamma(0) = \phi$ contains the initial state of the system (within an ellipsoid) and $\beta(t) \geq 0$ for all t. The parameter $\beta(t)$ enters in the construction of the bounding ellipsoid.

2. The controls u and outputs y are bounded by the ellipsoids described by the sets:

$$\Omega_u = \{u : u' \; (K\,\Gamma\,K')^{-1} u \leq 1\} \tag{9}$$

$$\Omega_y = \{y : y' \; (C\,\Gamma\,C')^{-1} y \leq 1\} \tag{10}$$

If the limits in controls and outputs are $umax(=\mid umin \mid$ because they are symmetrical bounds) and $ymax(=\mid ymin \mid)$ respectively, the controls and the outputs will respectively be given by

$$\mid u_i \mid \leq umax_i = \sqrt{T_i} \quad \text{for} \quad 1 \leq i \leq m \tag{11}$$

$$\mid y_i \mid \leq ymax_i = \sqrt{S_i} \quad \text{for} \quad 1 \leq i \leq n \tag{12}$$

Then, from (9) and (10), we obtain

$$K_i\;\Gamma\;K_i' \leq T_i \quad \text{for} \quad 1 \leq i \leq m \tag{13}$$

$$C_i\;\Gamma\;C_i' \leq S_i \quad \text{for} \quad 1 \leq i \leq n \tag{14}$$

It is not necessary for the bounds to be symmetrical. A more general formulation where the bounds are allowed to vary about a nominal point has been considered and is given in [11].

Note that the structure of this formulation has been interpreted in [5] as providing sufficient conditions for the stability and performance robustness of a closed loop system, measured in terms of minimization of a weighted sum of the state variables and subject to uncertainties in the closed loop matrix $A + B\,K$ and to white noise input disturbances. Their approach require that the designer chooses some matrices or parameters. In our case only the bounds to the inputs and outputs must be specified.

Also, the Lyapunov structure of equation (8) has been used to draw conclusions about the stability of the closed loop system [16,15,24] which in their cases is an end in itself, but in our case naturally arises as a consequence of imposing ellipsoidal constraints to the time-domain behavior of the system.

Single disturbance case

The solution to the single disturbance case has been presented in [13]. Here it is repeated for ease of reference when compared to the extension presented below. The solution to the multiple disturbances case is presented in [1].

The solution is found by means of an iterative procedure, when the following set of equations is simultaneously satisfied (under the assumption that the pair $(A + \beta I/2, B)$ is stabilizable and the pair $(A + \beta I/2, C'\sqrt{\Lambda_c})$ is detectable):

$$\left(A + \frac{\beta}{2}I\right)'\Lambda + \Lambda\left(A + \frac{\beta}{2}I\right) + C\,'\Lambda_c\,C = \Lambda\;B\;\Lambda_k^{-1}\;B'\;\Lambda \tag{15}$$

$$\left(A + BK + \frac{\beta}{2}I\right)'\Phi + \Phi\left(A + BK + \frac{\beta}{2}I\right) + \frac{G'\,G}{\beta} = 0 \tag{16}$$

$$K = -\Lambda_k^{-1}\;B'\Lambda \tag{17}$$

$$\beta^2 = \frac{trace(\Lambda\;GG')}{trace(\Lambda\;\Phi)}; \qquad \beta > 0 \tag{18}$$

$$\Lambda_k(diag(K\,\Phi\,K' - \frac{T}{Q})) = 0; \quad \Lambda_k > 0 \tag{19}$$

$$\Lambda_c(diag(C\,\Phi\,C' - \frac{S}{Q})) = 0; \quad \Lambda_c \geq 0 \tag{20}$$

$$trace(\Lambda_c\;S) + trace(\Lambda_k\;T) = 1 \tag{21}$$

LEB compensator design with measurement noise included

Until now, the design has proceeded with the LEB regulators and filters being developed separately [23,20]. The use of the separation principle allows a "separate but not independent" synthesis of the regulator and the filter in LQG designs [2]. However, a little of thought will indicate that the effect of a control law of the type $u = K\,x$ (with feedback of the real state variable x) will not be the same as the feedback $u = K\,\hat{x}_b = K\,(x - e)$ (with feedback of the estimated state variable $\hat{x}_b$) on the proposed control limits given by equation (11). The same considerations apply to the output limits. In this section the synthesis of an LEB compensator will be considered. Note that the design of the LEB regulator cannot be made unless the bounds for the measurement noise are known.

When there is no measurement noise, the generated control law is

$$u(t) = Kx(t) \tag{22}$$

but in the presence of a filter in the feedback path, the new control law will be

$$u_1(t) = K_1\,\hat{x}_b(t) = K_1\,(x(t) + e(t)) = u_2(t) + u_3(t) \tag{23}$$

where K_1 is the new modified LEB regulator gain, $\hat{x}_b$ is the estimated state variable, x is the real estate variable, e is the estimation error, u_2 is the control law when there are only plant noise (disturbances) and u_3 is the control law when there are no disturbances and there is measurement noise. In Figure 1 the structure of the feedback system is shown.

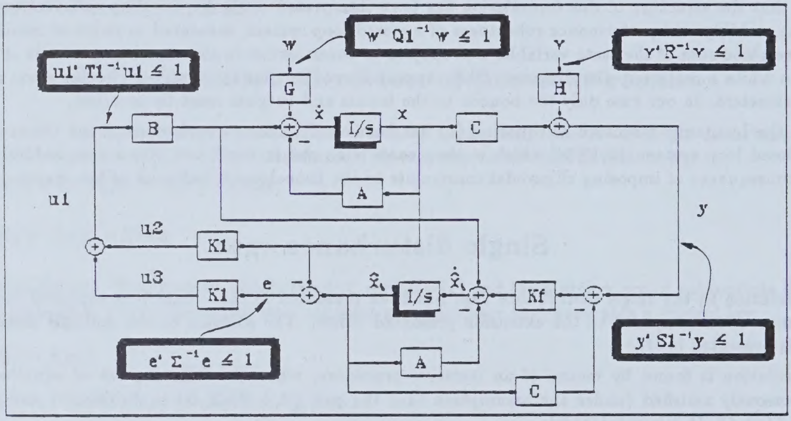


Figure 1: Structure of the plant and LEB compensator.

Using support functions for u_2 and u_3, applying a Holder inequality to the resulting expression and defining a new variable ϵ $(0 < \epsilon \le 1)$ the control u_1 can be defined such that u_{1b} is contained inside an ellipsoid $\Omega_{u_{1b}}$ which contains, in turn, the ellipsoid Ω_{u_1}, where

$$\Omega_{u_{1b}} : (u_{1b} : u'_{1b}\,T_1^{-1}\,u_{1b} \le 1) \tag{24}$$

$$T_1 = K_1 \left(\frac{\Gamma_1}{\epsilon} + \frac{\Sigma}{1-\epsilon} \right) K'_1. \tag{25}$$

K_1 and Γ_1 are obtained solving the regulator problem for $u_1 = K_1\,x_b$. Σ is the n x n matrix used to describe the ellipsoid that contains the estimation errors [20].

The bounds for the modified ith control (i=1,2,...,m) are now given by:

$$K_{1i} \left(\frac{\Gamma_1}{\epsilon} + \frac{\Sigma}{1-\epsilon} \right) K'_{1i} = T_{2(i,i)} + T_{3(i,i)} = T_{1(i,i)} \le T_{(i,i)} \tag{26}$$

The magnitude of the ith control $| u_{i max} |$ is less than or equal to $(| \sqrt{T_{2(i,i)}} | + | \sqrt{T_{3(i,i)}} |)$. In the presence of estimation errors, there will always be violations of the prescribed limits.

An analogous procedure can be followed for the output limits. In this case another new variable δ has to be defined, $(0 < \delta \leq 1)$. The maximum ith output excursion $| y_{i max} |$ will now be less or equal than $(| \sqrt{S_{2(i,i)}} | + | \sqrt{S_{3(i,i)}} |)$, where $S_2 = C\ \Gamma\ C' /\ \delta$, $S_3 = C\ R\ C' /\ (1\ -\ \delta)$ and R defines the ellipsoid that contains the measurement noise [20].

When ϵ and δ are incorporated in the previously considered iterative solution, the equations are slightly modified and optimal values of ϵ and δ result. The set of equations to be simultaneously solved, (27) to (38), is presented below. The regulator equations are:

$$(A + BK_1 + \frac{\beta}{2}I)'\Lambda + \Lambda(A + BK_1 + \frac{\beta}{2}I) + \frac{C'\ \Lambda_c\ C}{\delta} + \frac{K_1'\ \Lambda_k\ K_1}{\epsilon} = 0 \tag{27}$$

$$(A + BK_1 + \frac{\beta}{2}I)'\Phi + \Phi(A + BK_1 + \frac{\beta}{2}I) + \frac{G'\ G}{\beta} = 0 \tag{28}$$

$$K_1 = -\Lambda_k^{-1}B'\Lambda\Phi\left(\frac{\Phi}{\epsilon} + \frac{\Sigma}{Q_1(1-\epsilon)}\right)^{-1} \tag{29}$$

$$\beta^2 = \frac{trace(\Lambda\ G\ G')}{trace(\Lambda\ \Phi)}; \qquad \beta > 0 \tag{30}$$

$$\epsilon = \frac{1}{1 + \sqrt{\frac{trace(\Lambda_k\ K_1\ \Sigma\ K_1')}{trace(\Lambda_k\ K_1\ \Phi\ K_1')}}}; \qquad \epsilon > 0 \tag{31}$$

$$\delta = \frac{1}{1 + \sqrt{\frac{trace(\Lambda_c\ C\ R\ C')}{trace(\Lambda_c\ C\ \Phi\ C')}}}; \qquad \delta > 0 \tag{32}$$

$$\Lambda_k\left(diag\left(K_1(\frac{\Phi}{\epsilon} + \frac{\Sigma}{Q_1(1-\epsilon)})K_1' - \frac{T}{Q_1}\right)\right) = 0; \qquad \Lambda_k > 0 \tag{33}$$

$$\Lambda_c\left(diag\left(C(\frac{\Phi}{\delta} + \frac{R}{Q_1(1-\delta)})C' - \frac{S}{Q_1}\right)\right) = 0; \qquad \Lambda_c \geq 0 \tag{34}$$

$$trace\left(\Lambda_c(S - \frac{CRC'}{1-\delta})\right) + trace\left(\Lambda_k(T - \frac{K_1\ \Sigma\ K_1'}{1-\epsilon})\right) = 1 \tag{35}$$

The (steady-state) filter equations are:

$$A'\Sigma + \Sigma A + \frac{G'Q_1G}{\beta_f} - \rho_f\Sigma CR^{-1}C'\Sigma = 0 \tag{36}$$

$$K_f = \rho_f\Sigma C'R^{-1} \tag{37}$$

$$0 \leq \beta_f < \infty; \qquad 0 \leq \rho_f < \infty \tag{38}$$

As before, the pair $(A + \beta I/2, B)$ must be stabilizable and the pair $(A + \beta I/2, C'\sqrt{\Lambda_c})$ detectable):

As they should, these equations simplify to the no-noise case when R is null. In that case $\delta = \epsilon$=1, $K_1 = K$, $Q_1 = Q$.

The existence of these coupled Riccati-Lyapunov equations suggest that homotopic continuation methods could be used for their solution, as suggested in [10], that an interpretation of these results may be found in the light of LQG solutions with $H\infty$ bounds [4] and the "worst disturbance input" in $H\infty$ control problems [9] and that some relationship may exist with linear quadratic differential games [9].

The effect of the noise R on the system is to decrease the achieved control and output bounds and, thus, to decrease the allowed disturbances with respect to the no-noise case. The higher the contribution R, the smaller has to be the disturbance Q_1 until the specified control and output bounds are reached only due to the effect of the measurement noise. This effect can be seen in Figure 2, where the disturbance Q_1 was found from equations (33)-(41) for the system:

$$\begin{array}{lll} dx(t)/dt = -x(t) + 2u(t) + w(t); & & \mid w \mid = \sqrt{Q_1} \\ y(t) = x(t) + v(t); & \mid y \mid \leq 5; & \mid v \mid = \sqrt{R} \\ u(t) = K_1(x(t) + e(t)); & \mid u \mid \leq 6; & \mid e \mid = \sqrt{\Sigma} \end{array}$$

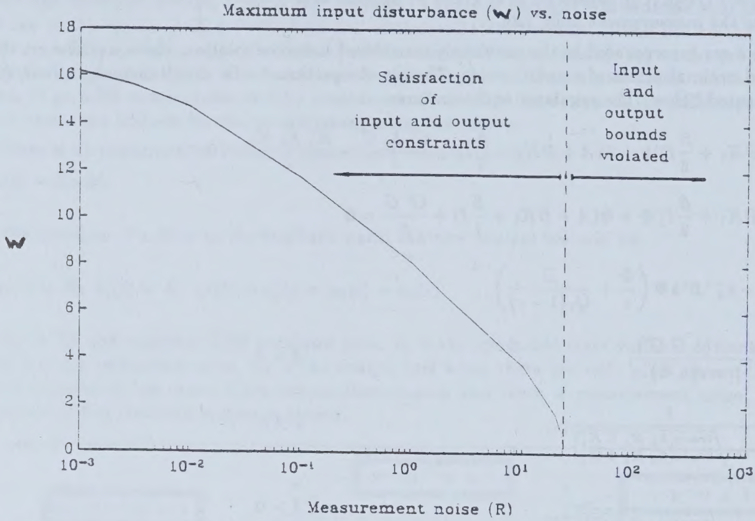


Figure 2: Input disturbances decrease to satisfy control and output bounds if measurement noise increases.

When there is no measurement noise in the system, there are no estimation errors ($\Sigma = 0$); the maximum allowable input disturbance is then $w = \sqrt{Q_1} = \sqrt{Q} = 17$.

Stability robustness of the LEB regulators to modelling errors

The optimization procedures to find the LEB regulator and filter gains are based in maximizing allowable input disturbances and minimizing estimation errors. For that reason these LEB designs should produce, when the conditions are satisfied, naturally more robust regulators and filters than other techniques that do not take disturbances into consideration from the start. At the same time, the LEB solutions have other benefits: fast response [23], no overshoots above the prescribed limits and no need for the designer to perform iterations. In this section we will show that we can obtain improved robustness margins with LEB designs.

In the last decade considerable attention has been given to robustness of control systems in the frequency and time domain, citing [16,15,8,21,17,5] to name a few. A straight comparison of these methods with the LEB design is not quite exact since, among other things, the description of variables may be different (stochastic versus bounded), the robustness measures could be obtained in other domains (frequency versus time) or defined in a different way.

With respect to modelling errors, we can find from [16] some results concerning the robustness of the closed-loop system

$$dx(t)/dt = Ax(t) + Bu(t) \tag{39}$$

with degree of stability $\beta/2$ to nonlinear perturbations $\Gamma(x(t), u(t))$. It is proved there that if in the perturbed system

$$dx(t)/dt = (A + \Delta_A)x(t) + (B + \Delta_B)u(t) \tag{40}$$

the 2-norms of the perturbation matrices Δ_A and Δ_B satisfy

$$\overline{\sigma}(\Delta_A) + \overline{\sigma}(\Delta_B)\, \overline{\sigma}(K) \leq \mu \tag{41}$$

where the scalar robustness measure μ is defined by

$$\mu = \frac{\underline{\sigma}(T_\beta) + \beta\, \underline{\sigma}(\Lambda_\beta)}{2\, \overline{\sigma}(\Lambda_\beta)} \tag{42}$$

and Λ_β is the solution to the Riccati equation (15), and

$$T_\beta = C'\, \Lambda_c\, C + \Lambda_\beta\, B\, \Lambda_k^{-1}\, B'\, \Lambda_\beta \tag{43}$$

then the closed-loop system

$$dx(t)/dt = (A + \Delta_A + (B + \Delta_B)K)x(t) + \Gamma(x(t), u(t)) \tag{44}$$

is stable. It is assumed that the pair (A,B) is controllable.

Applying these results to our LEB regulator, we may compare the stability robustness of the system with an LEB regulator and an LQ regulator. For a scalar case, with

$$dx(t)/dt = -x(t) + 2\, u(t) + w(t); \qquad y(t) = x(t), \tag{45}$$

with desired output bounds $\mid y \mid \leq 5$ and control bounds $\mid u \mid \leq 6$, the LEB solution is $K_{LEB} = -1.2$ ($u = K_{LEB} x$), $\Lambda_\beta = 3.4$, $\Phi = 0.0859$, $\Lambda_c = 3.4$, $\Lambda_k = 5.6665$, $\beta = 3.4$, and the LQ solution (with output weighting $Q = 1/5^2$ and control weighting $R = 1/6^2$) is $K_{LQ} = 0.8$ (for $u = -K_{LQ}\, x$) with Riccati equation solution s=0.0111.

Example: for the general case of LEB and LQ regulators for a scalar system (a,b,c), it can be easily proved that the change in robustness due to the use of a LEB regulator instead of a LQ regulator is

$$\Delta\mu = \beta + c^2 \frac{\Lambda_c}{\Lambda_\beta} - \frac{b^2(Q/R)}{a + \sqrt{a^2 + b^2(Q/R)}} \tag{46}$$

For the previous scalar example, if we introduce the parameters of the LQ Riccati equation (the Q and R weighting matrices) in (43) as unknowns and we assume that the LEB design is fixed, we may be able to find out the conditions that the LEB regulator has to satisfy to provide a better robustness measures than the LQ regulator. Doing so, we find that if $Q/R < 2.64$, then $\mu^{LEB} > \mu^{LQ}$. In our case $Q/R = 1.44$, so the LEB regulator is more robust to modelling errors than the LQ regulator.

Also, now from [15], we can directly compare the effect of LEB and LQ regulators on the μ_1 perturbation bound defined there, which is an improved version of the previous μ bound [16]:

$$\mu_1 = \frac{1}{\overline{\sigma}(P)} \tag{47}$$

$$A'P + PA' + 2I = 0 \tag{48}$$

In the scalar case, μ_1 is simply the absolute value of the closed loop system eigenvalue. In our scalar example, μ_1 is 3.4 for the LEB regulator and 2.6 for the LQ regulator showing that the first regulator provides more stability robustness. But more interesting than that is that, analytically, μ_1 can be shown to be equal to β. Consequently, the LEB solution immediately provides a bound to the stability robustness of the system. For the general multivariable case, it will be seen in the next section that the guaranteed stability robustness bound for LEB regulators for simultaneous row structured perturbations is $\beta/2$.

Example: application to flare control of the AWJSRA

The results of the previous section were applied to the longitudinal dynamics modes of the Augmentor Wing Jet STOL Research Aircraft (AWJSRA) [16,15,18] . The state variables and the controls are the following:

$$x = \left[\begin{array}{c} \delta v \\ \delta\gamma \\ \delta\theta \\ \delta q \\ \delta h \end{array}\right] \begin{array}{ll} \text{change in airspeed} & [ft/sec] \\ \text{change in flight-path angle} & [deg] \\ \text{change in pitch angle} & [deg] \\ \text{change in pitch rate} & [deg/sec] \\ \text{deviation from nominal altitude} & [ft] \end{array}$$

$$u \;=\; \begin{bmatrix} \delta e \\ \delta n \end{bmatrix} \begin{array}{ll} \text{change in elevator deflection} & [deg] \\ \text{change in nozzle angle} & [deg] \end{array}$$

The system matrices, expressed with the standard notation of equations (1) and (2), are:

$$A = \begin{bmatrix} -0.0547 & -0.2980 & -0.2639 & -0.0031 & 0 \\ 0.1600 & -0.4712 & 0.4661 & 0.0437 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 \\ 0.1752 & 0.1236 & -0.1236 & -1.3000 & 0 \\ -0.0174 & 1.9200 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0032 & -0.0943 \\ 0.0408 & 0.0224 \\ 0 & 0 \\ -1.1200 & -0.0800 \\ 0 & 0 \end{bmatrix} \; ; \qquad G = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

C is the identity matrix.
The LQ solution, with Q the identity matrix and $R = \mathrm{diag}([16\ 0;0\ 0.5])$, gives (for $u = -K_{LQ}\ x$)

$$K_{LQ} = \begin{bmatrix} -0.2546 & -0.6762 & -0.8109 & -0.5319 & -0.1924 \\ -2.6318 & -2.5443 & -2.2425 & -1.5543 & -0.9029 \end{bmatrix}$$

with $\mu^{LQ} = 0.0017$ [16] and $\mu_1^{LQ} = 0.0701$ [15]. The absolute value of the maximum real part of the closed loop eigenvalues is 0.1218.
The LEB solution is (for $u = K_{LEB}\ x$)

$$K_{LEB} = \begin{bmatrix} 0.1686 & 1.0945 & 1.4252 & 0.8971 & 0.2674 \\ 9.7354 & 11.9600 & 7.0435 & 4.1407 & 4.3502 \end{bmatrix}$$

with $\mu^{LEB} = 0.00001967$ [16] and $\mu_1^{LEB} = 0.1039$ [15]. Also, $\beta = 0.5241$ and the absolute value of the maximum real part of the closed loop eigenvalues is 0.5037.

If, instead of relying on calculated bounds we look for the actual bounds for each of the 25 elements of the closed loop system matrix $(A + B\ K)$ where K will be the LEB or LQ regulator, we will find out if those bounds are conservative and how does each regulator perform with respect to stability robustness. Doing so and defining Δ_+^{LEB} (Δ_-^{LEB}) the matrix whose ij component, when added (substracted) to the corresponding ij element of the closed loop system matrix $(A + B\ K_{LEB})$ defines a bound to the stability of the closed loop system, and Δ_+^{LQ} (Δ_-^{LQ}) for LQ regulator, where now the closed loop system matrix is $(A - B\ K_{LQ})$, we find that:

$$\Delta_-^{LEB} = \begin{bmatrix} \infty & \infty & 1.087 & 2.623 & 2.350 \\ 0.521 & 57.513 & 0.571 & 1.073 & O(44) \\ 0.516 & 56.985 & O(16) & 1.000 & 1.101 \\ 0.972 & 107.325 & 1.6\ 10^{32} & 1.2\ 10^{9} & 2.078 \\ 2.315 & 1.905 & 1.934 & 3.544 & O(18) \end{bmatrix}$$

$$\Delta_+^{LEB} = \begin{bmatrix} 0.898 & 2.697 & 6.845 & O(164) & 0.541 \\ O(149) & 0.565 & 3.030 & O(54) & 0.116 \\ O(65) & 1.330 & 0.966 & O(35) & 0.249 \\ O(64) & 3.097 & 3.433 & 2.372 & 0.554 \\ O(64) & 4.098 & 10.224 & O(66) & 0.248 \end{bmatrix}$$

$$\Delta_-^{LQ} = \begin{bmatrix} \infty & \infty & 0.352 & \underline{2.842} & 1.063 \\ 0.479 & \underline{O(17)} & 0.323 & \underline{1.296} & O(33) \\ 0.372 & 41.139 & O(14) & 1.000 & 0.498 \\ 0.582 & 18.854 & 1.5 10^{32} & \underline{6.9 10^{9}} & 0.734 \\ 1.760 & \underline{1.967} & 0.845 & \underline{4.298} & O(19) \end{bmatrix}$$

$$\Delta_{+}^{LQ} = \begin{bmatrix} 0.158 & 2.245 & 3.837 & O(41) & 0.210 \\ 0.583 & 0.482 & 2.357 & O(20) & 0.090 \\ \underline{O(116)} & 0.832 & 0.632 & \underline{O(36)} & 0.168 \\ 6.424 & 1.641 & 1.610 & 1.769 & 0.318 \\ 0.727 & 3.970 & 7.819 & O(64) & 0.202 \end{bmatrix}$$

Note that:

- in 41 out of 50 possible cases, the stability bounds provided by the LEB regulator are better than those given by the LQ regulator.
- the bounds given in [15] do not seem to be very conservative for this particular example: 0.1039 and 0.0701 for LEB and LQ regulators respectively vs. 0.116 and 0.090, the minimum stability bound (for element $(A + B\ K)_{25}$). The bound given in [18] is 0.0360.
- the bounds given in [16] do seem very conservative for this example: 0.00001967 and 0.0017 for LEB and LQ regulators respectively vs. 0.116 and 0.090.

In these previous matrices, ∞ stands for elements larger than 10^{308} and the underlined elements are the few situations where a LQ regulator could give better results than an LEB regulator.

From this collection of perturbation bounds, it can be seen that, in most cases, LEB bounds are greater than LQ bounds, providing, in addition, guaranteed time domain bounded behavior for controls and outputs. Note, however, that these results depend on the selected values of Q and R. Changing them would certainly modify the results but finding the Q and R matrices that would simultaneously guarantee time domain bounded behavior and good stability robustness properties for LQ will certainly take some time, while LEB results were designed specifically to do that.

Another techniques worthwhile to be investigated with respect to parameter variations are those referenced in [3,22,7] although no guaranteed control and output bounds are provided by them and [6,19] althogh now the implementation may not be straightforward.

The reason that this technique gives such good "structured" stability robustness results stems from the fact that a row-structured perturbation of magnitude $\beta/2$ is added to the diagonal of the system matrix A, and the design procedure synthesizes a regulator for this perturbed system. This effect can be seen in equations (8), (15), (16), (27) and (28). Consequently, all the closed-loop eigenvalues will be at the left of the $-\beta/2$ line in the $s-$plane.

General comments about the applicability of LEB compensators

In this section we will comment issues related to the robustness of LEB compensators, including the presence of modelling errors, the effect of gain and phase margin, robustness recovery and roll off effects [1,16,14].

In reference [14] some comments about regulators with a prescribed degree of stability can be found, and for some time their relationship with LQG designs [23] has been known. In [14] the following observations were made:

- in the frequency domain, robustness improves for certain multiplicative errors for increasing values of β, and gain and phase margin depend on the system in consideration.
- robustness recovery is affected by β, and too high values imply that Kwakernaak's [12] or Doyle's [8] procedures may not recover desired robustness measures.
- large values of β extend the bandwidth of the LEB system into frequencies where the effect of unmodelled or unknown dynamics may be important.

In the previous comments it was considered that the β of the system can be modified, which is the case of regulators with a prescribed degree of stability. But in our case, the parameter β is just needed to construct a bounding ellipsoid and, if the results given by the algorithm are accepted, it is not a designer's choice. Thus the performance of the system has to be balanced with the benefits obtained from high β's (if it is necessary to use a high value of that parameter), namely, the high speed of response. However, this choice is ever present: large loop gains increase speed of response at the same time that they dangerously increase the bandwidth of the system making it more sensitive to unknown dynamic effects.

Anyway, nothing precludes from specifying *apriori* the β of the system at the same time that bounds for controls and outputs are given. In this case the equations (15)-(21) and (27)-(38) will have to be modified because β will be fixed.

From the results presented in the previous section and here, the designer now has a way of assessing how the addition of noise decreases the amount of allowable outside input disturbances (i.e. Q_1 has to decrease) and how the perturbations in the A and B matrices affect the closed loop stability. It is interesting to note that the effect of perturbations in the A matrix do not affect the achievable control and output bounds, they simply reduce or increase the allowable input disturbances WITHOUT changing the feedback gains. This behavior can easily be seen from the scalar example, where the gain is the quotient between the desired control bounds $\mid \pm 6 \mid$ and output bounds $\mid \pm 5 \mid$. Thus $K_{LEB} = -1.2$, independently of the numerical values of the system matrices. A change in the system matrices modifies the magnitude of the input disturbance and the closed loop eigenvalues (until the system is closed loop unstable), not the regulator gains. This is a consequence of working with bounds and constraining the regulator to be a mapping between the output and input bounds.

Conclusions

In this paper, extensions in the design of LEB regulators that allow the design of control and input bounded MIMO LTI when measurement noise is included have been presented. At the same time, new results concerning the robustness to plant, measurement and modelling errors were shown. These results indicate that

1. input disturbances decrease to accommodate the effect of measurement noises while the basic control and input bounded behavior is maintained (performance robustness of the LEB compensator),
2. an LEB regulator may allow more modelling errors of the system matrices A and B compared to an LQ regulator before the closed loop system becomes unstable (stability robustness of the LEB regulator).

MIMO and a scalar examples were given to illustrate these effects.

References

[1] C. Antonini. *Unification and extensions in the design of Linear-Ellipsoidal-Bounded regulators and filters.* PhD thesis, Buenos Aires Institute of Technology, 1985.

[2] M. Athans. The role and use of the stochastic LQG problem in control systems design. *IEEE TAC*, December 1971.

[3] R. Badr, M. Hassan, J. Bernussou, and A. Bilal. Time-domain robustness and performance optimization of LQ control problems. *IEE Proceedings*, July 1988.

[4] D. Bernstein and W. Haddad. LQG control with an H∞ performance bound: a Riccati equation approach. *IEEE-TAC*, March 1989.

[5] D. Bernstein and W. Haddad. Robust stability and performance analysis for linear dynamic systems. *IEEE-TAC*, July 1989.

[6] S. Boyd. A new CAD method and associated architectures for linear controllers. *IEEE-TAC*, March 1988.

[7] B. Chen and T. Dong. LQG optimal control system design under plant perturbation and noise uncertainty: a state-space approach. *Automatica*, May 1989.

[8] J. Doyle. Robustness with observers. *IEEE-TAC*, August 1979.

[9] J. Doyle, K. Glover, P. Khargonekar, and B. Francis. State-space solutions to standard H_2 and $H\infty$ control problems. *IEEE-TAC*, August 1989.

[10] W. Greeley and D. Hyland. Reduced order compensation: LQG reduction versus optimal projection using a homotopic continuation method. In *IEEE Conference on Decision and Control*, pages 742–747, Los Angeles, CA, 1987.

[11] L. Gulko. *Set theoretic tracking.* Master's thesis, Department of Mechanical Engineering, MIT, 1981.

[12] H. Kwakernaak. Optimal low-sensitivity linear feedback systems. *Automatica*, May 1969.

[13] S. Neghadaripour. *A set theoretic control algorithm.* Master's thesis, Department of Mechanical Engineering, MIT, 1980.

[14] P. Ng. *On regulators with a prescribed degree of stability.* Master's thesis, Department of Electrical Engineering, MIT, 1981.

[15] R. Patel and M. Toda. Quantitative measures of robustness for multivariable systems. In *Journals of the ACC*, 1980.

[16] R. Patel, M. Toda, and B. Sridhar. Robustness of LQ state feedback designs in the presence of system uncertainty. In *Journals of the ACC*, pages 1668–1673, 1977.

[17] I. Petersen. Disturbance attenuation and H_∞ optimization: a design method based on the algebraic Riccati equation. *IEEE-TAC*, May 1987.

[18] D. Petkovski. Improved time-domain stability robustness measures for linear regulators. *AIAA-JGDC*, July-August 1989.

[19] E. Polak and S. Salcudean. On the design of linear multivariable feedback systems via constrained nondifferentiable optimization in h_∞ spaces. *IEEE-TAC*, March 1989.

[20] F. Schweppe. *Uncertain Dynamic Systems.* Prentice Hall, 1973.

[21] D. Siljak. Parameter space methods for robust control design: a guided tour. *IEEE-TAC*, July 1989.

[22] D. Tahk and J. Speyer. Parameter robust LQG design synthesis with flexible structure control applications. *AIAA Journal of Guidance, Control and Dynamics*, July-August 1989.

[23] P. Usoro, F. Schweppe, L. Gould, and D. Wormley. A Lagrange approach to set-theoretic control synthesis. *IEEE-TAC*, April 1982.

[24] R. Yedavalli. Robust control design for aerospace applications. *IEEE-TAES*, May 1989.

SOME APPLICATIONS OF INTERVAL ANALYSIS METHODS TO CIRCUITS AND SYSTEMS*

A. N. Michel H. F. Sun
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

1. INTRODUCTION.

In an attempt to account for finite word length effects in digital computers, Moore [1], [2], and others introduced interval arithmetic and interval analysis methods. Subsequently, these methods have been used in a variety of applications, including solutions of nonlinear circuit equations[6], two-point boundary value problems[8], linear programming[9], computation of eigenvalues[10], worst-case analysis of linear circuits[7], determination of envelopes of the solutions of a linear system endowed with a single parameter belonging to an interval[3]-[5] and [13], and the like.

The changes in the performance of a dynamical system due to the changes in system parameters are of great practical importance in engineering analysis and design. Such perturbations of parameters may model the effects of uncertainties in manufacturing tolerances, "aging" of components, environmental causes and the like. Because of these uncertainties, the values of the parameters of a given dynamical system may frequently be viewed as belonging to suitable intervals. In doing so, the system description is then phrased in terms of differential equations, or difference equations involving parameters that belong to appropriate intervals.

The results in [3]-[5] and [13] address continuous time, finite dimensional, linear dynamical systems endowed with one uncertain parameter belonging to a specified interval. What we present in this parper are some new general fundamental interval analysis results which enable us to estimate solution bounds for initial-value problems which (a) are endowed with more than one parameter belonging to more than one interval, (b) need not be linear and (c) are applicable to continuous-time and to discrete-time systems.

2. NOTATION.

Let V and W be arbitrary sets. Then $V \cup W$, $V \cap W$, $V - W$ and $V \times W$ denote the union, intersection, difference and Cartesian product of V and W, respectively. If V is a subset of W, we write $V \subset W$. We say V and W are equal, written as $V = W$, if $V \subset W$ and $W \subset V$. For any x belonging to V, we write $x \in V$ and we call x a point or an element of V. If f is a mapping from V into W, we write $f : V \to W$ and $f(U) = \{f(x) \in W \mid x \in U\}$ for $U \subset V$, and $f^{-1}(y) = \{x \in V \mid f(x) = y\}$ for $y \in W$. Let R denote the set of real numbers and let R^+ denote the set of nonnegative real numbers.

An *interval*, I, is defined to be a subset of the real numbers of the form,

$$I = \{x \in R \mid a \leq x \leq b \text{ with } a, b \in R\}.$$

We denote I as $[a, b]$. If $b = a$, the interval $[a, a]$ is called a degenerate interval. Two intervals I and J are said to be equal if they are equal in the set theoretic sense. That is, $I = J$ if and only if $a = c$ and $b = d$, where $I = [a, b]$ and $J = [c, d]$.

---

* Supported by the National Science Foundation under grant ECS 88-02924.

Let $\mathcal{J}$ be the set of all intervals on R, and let $\mathcal{J}^{(k)}$ be the set of the k Cartesian product of $\mathcal{J}$. For a fixed element I in $\mathcal{J}$, define $\mathcal{J}_I$, a subset of $\mathcal{J}$, as

$$\mathcal{J}_I = \{J \in \mathcal{J} \mid J \subset I\}.$$

Similarly, we can define a subset,$\mathcal{J}_{\mathbf{I}}^{(k)}$, of $\mathcal{J}^{(k)}$ for a fixed element $\mathbf{I} = (I_1, I_2, \cdots, I_k)$ in $\mathcal{J}^{(k)}$ as

$$\begin{aligned}\mathcal{J}_{\mathbf{I}}^{(k)} &= \{\mathbf{J} = (J_1, J_2, \cdots, J_k) \in \mathcal{J}^{(k)} \mid J_1 \subset I_1, J_2 \subset I_2, \cdots, J_k \subset I_k\} \\ &= \mathcal{J}_{I_1} \times \mathcal{J}_{I_2} \times \cdots \times \mathcal{J}_{I_k}.\end{aligned}$$

3. PRELIMINARIES.

For any two elements I and J in $\mathcal{J}$, define the interval arithmetic operations as

$$I * J = \{x * y \mid x \in I, y \in J\},$$

where $*$ is one of the symbols $+, -, \cdot$, and $/$, except that we do not define I/J if $0 \in J$. In terms of the corresponding real interval endpoints, for $I = [a, b]$ and $J = [c, d]$, we define

$$\begin{aligned} I + J &= [a, b] + [c, d] = [a + c, b + d] \\ I - J &= [a, b] - [c, d] = [a - d, b - c] \\ I \cdot J &= [a, b] \cdot [c, d] = [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\ \text{and } I/J &= [a, b]/[c, d] = [\min\{\tfrac{a}{d}, \tfrac{a}{c}, \tfrac{b}{d}, \tfrac{b}{c}\}, \max\{\tfrac{a}{d}, \tfrac{a}{c}, \tfrac{b}{d}, \tfrac{b}{c}\}] \text{ provided that } 0 \notin [c, d]. \end{aligned}$$

For given interval $I = [a, b]$ in $\mathcal{J}$, define the width of I as

$$W(I) = |(b - a)|.$$

If I is a degenerate interval, then $W(I) = 0$.

A real-valued function $D: X \times X \rightarrow R^+$ is called a metric if $D(x, y)$ satisfies the following properties for all $x, y, z \in X$,

(i) $D(x, y) = D(y, x)$;
(ii) $D(x, y) \geq 0$ and $D(x, y) = 0$ if and only if $x = y$;
and (iii) $D(x, z) + D(z, y) \leq D(x, y)$,

where X is an arbitrary non-empty sets. The pair of objects $\{X, D\}$ is called a metric space. The above properties are respectively called *symmetry*, *positivity* and *strict positivity* and the *triangle inequality*.

The workable abstract mathematical structure which will be used in the current paper is that of metric space. To make $\mathcal{J}$ into a metric space, we define a real-valued function $\rho_1: \mathcal{J} \times \mathcal{J} \rightarrow R^+$ as

$$\rho_1(I, J) = \max\{|a - c|, |b - d|\}, \text{ for all } I = [a, b],\ J = [c, d] \in \mathcal{J}.$$

where $|x|$ is the absoulate value of x. It is clear that ρ_1 satisfies the *symmetry*, *positivity* and *strict positivity* properties. And, it can be easily shown that ρ_1 satisfies the *triangle inequality* since $|a - e| \leq |a - c| + |c - e|$ for aribitrary real numbers a, c and e.

Therefore, $\{\mathcal{J}, \rho_1\}$ is a metric space. Similarly, we can define a real-valued function $\rho_2: \mathcal{J}^{(2)} \times \mathcal{J}^{(2)} \rightarrow R^+$ for any $\mathbf{I} = (I_1, I_2)$ and $\mathbf{J} = (J_1, J_2)$ in $\mathcal{J}^{(2)}$ by

$$\rho_2(\mathbf{I}, \mathbf{J}) = \max\{\rho_1(I_1, J_1), \rho_1(I_2, J_2)\},$$

where ρ_1 is defined above.

It is clear that $\rho_2(\mathbf{I}, \mathbf{J}) = \rho_2(\mathbf{J}, \mathbf{I})$ for any $\mathbf{I} = (I_1, I_2)$, $\mathbf{J} = (J_1, J_2) \in \mathcal{J}^{(2)}$ since $\rho_1(I_1, J_1) = \rho_1(J_1, I_1)$ and $\rho_1(I_2, J_2) = \rho_1(J_2, I_2)$. It is also clear that $\rho_2(\mathbf{I}, \mathbf{J}) \geq 0$ for any $\mathbf{I}, \mathbf{J} \in \mathcal{J}^{(2)}$ and $\rho_2(\mathbf{I}, \mathbf{J}) = 0$ if and only if $\mathbf{I} = \mathbf{J}$ (i.e., $I_1 = J_1$ and $I_2 = J_2$) since $\{\mathcal{J}, \rho_1\}$ is a metric space. It is also easy to show that ρ_2 satisfies the *triangle inequality* since $\{\mathcal{J}, \rho_1\}$ is a metric space. Thus, $\{\mathcal{J}^{(2)}, \rho_2\}$ is a metric space.

Remark : In general, we can define a real-valued function, $\rho_k: \mathcal{J}^{(k)} \times \mathcal{J}^{(k)} \rightarrow R^+$ as

$$\rho_k(\mathbf{I}, \mathbf{J}) = \max_i \rho_1(I_i, J_i)$$

for any two elements $\mathbf{I} = (I_1, I_2, \cdots, I_k)$ and $\mathbf{J} = (J_1, J_2, \cdots, J_k) \in \mathcal{J}^{(k)}$ and for an arbitrary positive integer k. It is also easy to verify that $\{\mathcal{J}^{(k)}, \rho_k\}$ is a metric space for any positive integer k. At this point, we are only interested in the cases $k = 1$ and 2.

Thus far, we have introduced two metric spaces $\{\mathcal{J}, \rho_1\}$ and $\{\mathcal{J}_I, \rho_1\}$. To employ these metric spaces, we have to establish some of their properties, e.g, completeness and compactness[11]. To accomplish this, we first prove the corresponding properties for certain submetric spaces, $\{X_1, D\}$ and $\{X_2, D\}$, of the two dimensional Euclidean space. Then we establish an isometry between spaces $\{\mathcal{J}, \rho_1\}$, $\{\mathcal{J}_I, \rho_2\}$, and the submetric spaces $\{X_1, D\}$, $\{X_2, D\}$, respectively.

Let $X = R^2$, and let $D: X \times X \rightarrow R^+$ be defined as

$$D(\mathbf{x}, \mathbf{y}) = \max\{|x_1 - y_1|, |x_2 - y_2|\}$$

for every $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2) \in X$. Clearly, $\{X, D\}$ is a complete metric space[11, pp.292]. Consider a subset of X, $X_1 = \{(x_1, x_2) \in X \mid x_1 \leq x_2\}$. It can be easily shown that X_1 is closed in the topology induced by the metric space $\{X, D\}$. Hence, $\{X_1, D\}$ is a complete metric space since $\{X, D\}$ is complete[16, pp.127].

Define a mapping $f: \mathcal{J} \rightarrow X_1$, by $f([a, b]) = (a, b)$ for any element in $\mathcal{J}$, $I = [a, b]$. It can be easily verified that f is an isometry and $\{\mathcal{J}, \rho_1\}$ and $\{X_1, D\}$ are isometric. Since $\{X_1, D\}$ is complete and completeness is preserved under an isometry[14, pp.117], $\{\mathcal{J}, \rho_1\}$ is a complete metric space.

Now, consider another subset of X, $X_2 = \{(x_1, x_2) \in X \mid a \leq x_1 \leq x_2 \leq b\}$ for a given arbitrary $a, b \in R$. It can be easily shown that X_2 is closed in the topology induced by the metric space $\{X, D\}$. Hence, $\{X_2, D\}$ is a complete metric space since $\{X, D\}$ is complete[16, pp.127]. Clearly, X_2 is also bounded in the topology induced by the metric space $\{X, D\}$. Therefore, $\{X_2, D\}$ is a compact metric space since X_2 is a closed and bounded subset of $X = R^2$[15,pp. 56].

Define a mapping $f: \mathcal{J}_I \rightarrow X_2$ as $f([c, d]) = (c, d)$ for all $J = [c, d] \in \mathcal{J}_I$. It also can be easily verified that f is an isometry and $\{X_2, D\}$ and $\{\mathcal{J}_I, \rho_1\}$ are isometric. Therefore, $\{\mathcal{J}_I, \rho_1\}$ is a complete metric space since $\{X_2, D\}$ is complete[14, pp.117].

Since $\{\mathcal{J}_I, \rho_1\}$ and $\{X_2, D\}$ are isometric and $\{X_2, D\}$ is compact, $\{\mathcal{J}_I, \rho_1\}$ is a compact metric space [16, pp.126]. Therefore, $\{\mathcal{J}_I, \rho_1\}$ is a complete and compact metric space.

By viewing the metric space $\{\mathcal{J}^{(2)}, \rho_2\}$ ($\{\mathcal{J}_{\mathbf{I}}^{(2)}, \rho_2\}$) as a product space of $\{\mathcal{J}, \rho_1\}$ and $\{\mathcal{J}, \rho_1\}$ ($\{\mathcal{J}_{I_1}, \rho_1\}$ and $\{\mathcal{J}_{I_2}, \rho_1\}$), we have the following results for the metric spaces $\{\mathcal{J}^{(2)}, \rho_2\}$ and $\{\mathcal{J}_{\mathbf{I}}^{(2)}, \rho_2\}$ [16, pp.265].

Proposition 3.1 : $\{\mathcal{J}^{(2)}, \rho_2\}$ is a complete metric space.

Proposition 3.2 : $\{\mathcal{J}_{\mathbf{I}}^{(2)}, \rho_2\}$ is a complete and compact metric space for an arbitrary element $\mathbf{I} = (I_1, I_2) \in \mathcal{J}^{(2)}$.

Remark : Henceforth, the metric space $\{\mathcal{J}_{\mathbf{I}}^{(2)}, \rho_2\}$ is understood as a metric subspace of $\{\mathcal{J}^{(2)}, \rho_2\}$ for the underlying element $\mathbf{I} = (I_1, I_2)$ in $\mathcal{J}^{(2)}$.

Let,

$$\mathcal{F} = \left\{ f \mid f\colon \mathcal{J}_{\mathbf{I}}^{(2)} \to \mathcal{J}, f \text{ is a continuous function on } \mathcal{J}_{\mathbf{I}}^{(2)} \right\}.$$

Remark : The preceding definition of $\mathcal{F}$ can readily be extended to functions whose domain of definition is the space $\{\mathcal{J}^{(k)}, \rho_k\}$ where k is a fixed positive integer ($\mathcal{J}^{(k)}$ and ρ_k were defined earlier). Note that, since $\{\mathcal{J}_{\mathbf{I}}^{(2)}, \rho_2\}$ is compact, the function $f \in \mathcal{F}$ is uniformly continuous on $\mathcal{J}_{\mathbf{I}}^{(2)}$.

For any $f, g \in \mathcal{F}$, we define a distance function between f and g, $\mu\colon \mathcal{F} \times \mathcal{F} \to R^+$, as,

$$\mu(f,g) = \sup_{(K_1,K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}} \rho_1 \left(f(K_1,K_2), g(K_1,K_2) \right).$$

Since $\{\mathcal{J}, \rho_1\}$ is a metric space, it is clear that $\mu(f,g) \geq 0$ for any f and g belonging to $\mathcal{F}$ and $\mu(f,g) = 0$ if and only if $f(K_1,K_2) = g(K_1,K_2)$ for all $(K_1,K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}$, i.e., if and only if $f = g$. Also, $\mu(f,g) = \mu(g,f)$ for all $f, g \in \mathcal{F}$. We need to show that μ satisfies the *triangle inequality*. For any f, g and $h \in \mathcal{F}$,

$$\begin{aligned}
\mu(f,h) &= \sup_{(K_1,K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}} \rho_1 \left(f(K_1,K_2), h(K_1,K_2) \right) \\
&\leq \sup_{(K_1,K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}} \left[\rho_1 \left(f(K_1,K_2), g(K_1,K_2) \right) + \rho_1 \left(g(K_1,K_2), h(K_1,K_2) \right) \right] \\
&\leq \sup_{(K_1,K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}} \rho_1 \left(f(K_1,K_2), g(K_1,K_2) \right) + \\
&\qquad + \sup_{(K_1,K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}} \rho_1 \left(g(K_1,K_2), h(K_1,K_2) \right) \\
&\leq \mu(f,g) + \mu(g,h).
\end{aligned}$$

Therefore, μ is a metric and $\{\mathcal{F}, \mu\}$ is a metric space. In addition, it can be easily shown that $\{\mathcal{F}, \mu\}$ is a complete metric space.

4. MAIN RESULTS.

Before stating the main results which constitute generalizations to existing results given in [3]-[5] and [13], we will give the definition of the united extensionf or any arbitrary continuous interval function defined on $\mathcal{J}_{\mathbf{I}}^{(2)}$ and we will give the definition of rational interval function.

Definition 4.1

For any $f \in \{\mathcal{F}, \mu\}$, $\mathbf{K} = (K_1, K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}$, define the real valued functions $f^L(\cdot)$ and $f^R(\cdot)$ by

$$f(\mathbf{K}) \overset{\triangle}{=} [f^L(\mathbf{K}), f^R(\mathbf{K})].$$

Definition 4.2 (United extension).

For any $f \in \{\mathcal{F}, \mu\}$, $\mathbf{K} = (K_1, K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}$, define the *united extension* of f as,

$$\bar{f}(\mathbf{K}) \overset{\triangle}{=} \bigcup_{x \in K_1, y \in K_2} f(([x, x], [y, y])).$$

Definition 4.3 (Rational interval function).

$f(J_1, J_2, \cdots, J_k)$ is said to be a *rational* expression in the interval variables $J_1, J_2, \cdots, J_k$, if it is a finite combination of $J_1, J_2, \cdots, J_k$ and a finite set of constant intervals with interval arithmetic operations. A function $f\colon \mathcal{J}_{I_1} \times \mathcal{J}_{I_2} \times \cdots \times \mathcal{J}_{I_k} \to \mathcal{J}$ is called a *rational interval function* if there is a rational expression in the interval variables $J_1, J_2, \cdots, J_k$ which represents the values $f(J_1, J_2, \cdots, J_k)$ of f.

Remark : Henceforth, when using the term rational interval function, we have in mind the case for $k = 2$.

The principal result of this paper is given in Theorem 4.1. To prove this theorem, we need to establish the auxiliary results given in Proposition 4.1- Proposition 4.8 and Corollary 4.1 and 4.2. Due to space limitations, the proofs of these results will have to be omitted. They will be presented elsewhere.

Proposition 4.1 : For arbitrary $f \in \{\mathcal{F}, \mu\}$ and $\mathbf{K} = (K_1, K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}$, the united extension of f is given by

$$\bar{f}(\mathbf{K}) \overset{\triangle}{=} \bigcup_{x \in K_1, y \in K_2} f(([x, x], [y, y])) = [\mathrm{p}(\mathbf{K}), \mathrm{q}(\mathbf{K})] \in \mathcal{J}$$

where

$$\mathrm{p}(\mathbf{K}) = \inf_{x \in K_1, y \in K_2} f^L(([x, x], [y, y])),$$

and

$$\mathrm{q}(\mathbf{K}) = \sup_{x \in K_1, y \in K_2} f^R(([x, x], [y, y])).$$

Proposition 4.2 : For an arbitrary $f \in \{\mathcal{F}, \mu\}$, $\bar{f} \in \{\mathcal{F}, \mu\}$.

Proposition 4.3 : The rational interval functions belong to $\{\mathcal{F}, \mu\}$.

Proposition 4.4 : For any rational interval function $f \in \{\mathcal{F}, \mu\}$ and arbitrary $\mathbf{K} = (K_1, K_2) \in \mathcal{J}_{\mathbf{I}}^{(2)}$,

$$f(\mathbf{K}) \supset \bar{f}(\mathbf{K}) \overset{\triangle}{=} \cup_{x \in K_1, y \in K_2} f(([x, x], [y, y])).$$

Corollary 4.1 : For any rational interval function $f \in \{\mathcal{F}, \mu\}$ and arbitrary $\mathbf{L} \in \mathcal{J}_{\mathbf{I}}^{(2)}$, if $\mathbf{K} \in \mathcal{J}_{\mathbf{I}}^{(2)}$ and $\mathbf{L} \supset \mathbf{K}$, then $f(\mathbf{L}) \supset f(\mathbf{K}) \supset \bar{f}(\mathbf{K})$.

Proposition 4.5 : Let $\{f_n\} \to f$ be an arbitrary Cauchy sequence in $\{\mathcal{F}, \mu\}$. Then $\{\bar{f}_n\}$ is a Cauchy sequence in $\{\mathcal{F}, \mu\}$ converging uniformly to $\bar{f} \in \{\mathcal{F}, \mu\}$.

Proposition 4.6 : Let $\{f_n\} \to f$ be an arbitrary Cauchy sequence of rational interval functions in $\{\mathcal{F}, \mu\}$. Then for each $\mathbf{K} \in \mathcal{J}_{\mathbf{I}}^{(2)}$, $f(\mathbf{K}) \supset \bar{f}(\mathbf{K})$.

Corollary 4.2 : If $\{f_n\} \to f$ is an arbitrary Cauchy sequence of rational interval functions in $\{\mathcal{F}, \mu\}$, then for each $\mathbf{L} \in \mathcal{J}_{\mathbf{I}}^{(2)}$, if $\mathbf{K} \in \mathcal{J}_{\mathbf{I}}^{(2)}$ and $\mathbf{L} \supset \mathbf{K}$,

$$f(\mathbf{L}) \supset f(\mathbf{K}) \supset \bar{f}(\mathbf{K}).$$

Proposition 4.7 : Let $f \in \{\mathcal{F}, \mu\}$ have the property that for any $\mathbf{K} \in \mathcal{J}_{\mathbf{I}}^{(2)}$, $f(\mathbf{K}) \supset \bar{f}(\mathbf{K})$. Let $\mathbf{K} \triangleq ([c_1, c_{n+1}], [d_1, d_{m+1}])$ and $\mathbf{K}^{ij} \triangleq ([c_i, c_{i+1}], [d_j, d_{j+1}]) \in \mathcal{J}_{\mathbf{I}}^{(2)}$, with $c_i \leq c_{i+1}$ and $d_j \leq d_{j+1}$, for $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, m$. Then,

$$\bigcup_{i=1}^{n} \bigcup_{j=1}^{m} f(\mathbf{K}^{ij}) \supset \bar{f}(\mathbf{K}) \quad \text{and} \quad \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} f(\mathbf{K}^{ij}) \in \mathcal{J}.$$

Proposition 4.8 : Let $\{f_k\} \to f$ be a Cauchy sequence of rational interval functions in $\{\mathcal{F}, \mu\}$. For any $\mathbf{K} \in \mathcal{J}_{\mathbf{I}}^{(2)}$, let $\mathbf{K} \triangleq ([c_1, c_{n+1}], [d_1, d_{m+1}])$ and $\mathbf{K}^{ij} \triangleq ([c_i, c_{i+1}], [d_j, d_{j+1}]) \in \mathcal{J}_{\mathbf{I}}^{(2)}$, with $c_i \leq c_{i+1}$ and $d_j \leq d_{j+1}$, for $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, m$. Then,

$$f_k(\mathbf{K}) \supset \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} f_k(\mathbf{K}^{ij}) \supset \bar{f}_k(\mathbf{K}), \qquad k = 1, 2, \cdots$$

and this relation converges to the relation

$$f(\mathbf{K}) \supset \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} f(\mathbf{K}^{ij}) \supset \bar{f}(\mathbf{K}).$$

Theorem 4.1 : Let $\{f_k\} \to f$ be any Cauchy sequence of rational interval functions in $\{\mathcal{F}, \mu\}$. For any $\mathbf{K} \in \mathcal{J}_{\mathbf{I}}^{(2)}$, let $\mathbf{K} \triangleq ([a, b], [c, d])$ and let

$$\mathbf{K}_{nm}^{ij} \triangleq \left(\left[\frac{(n-i+1)a + (i-1)b}{n}, \frac{(n-i)a + ib}{n} \right], \left[\frac{(m-j+1)c + (j-1)d}{m}, \frac{(m-j)c + id}{m} \right] \right),$$

$i = 1, 2, \cdots, n$, $j = 1, 2, \cdots, m$ and $n, m = 1, 2, \cdots$. Then

$$f(\mathbf{K}) \supset \lim_{n \to \infty} \lim_{m \to \infty} \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} f(\mathbf{K}_{nm}^{ij}) = \bar{f}(\mathbf{K}).$$

5. EXAMPLES.

We consider second order digital filters in direct form as depicted in figure 1, where $U(k)$ is the input, a and b are parameters belonging to two intervals, say $A = [a_1, a_2]$ and $B = [b_1, b_2]$(see figure 3). In order to aovid confusion, we denote the state varaibles and output as $X_1(k, a, b)$, $X_2(k, a, b)$ and $Y(k, a, b)$ respectively, instead of the usual $X_1(k)$, $X_2(k)$ and $Y(k)$ when using the fixed parameters a and b. Then, for the fixed parameters a and b, the state equation is written as:

$$\begin{aligned} X_1(k+1, a, b) &= X_2(k, a, b) \\ X_2(k+1, a, b) &= aX_1(k, a, b) + bX_2(k, a, b) + U(k) \qquad \text{for } k = 1, 2, \cdots, \end{aligned}$$

and the output equation is

$$Y(k, a, b) = X_2(k, a, b) \qquad \text{for } k = 1, 2, \cdots.$$

From Definition 4.2, we have the united extension of the state variables and the output, at the time instance k as:

$$\begin{aligned} \bar{X}_1(k, A, B) &= \bigcup_{a\in A} \bigcup_{b\in B} X_1(k, [a, a], [b, b]), \\ \bar{X}_2(k, A, B) &= \bigcup_{a\in A} \bigcup_{b\in B} X_2(k, [a, a], [b, b]) \end{aligned}$$

and

$$\bar{Y}(k, A, B) = \bigcup_{a\in A} \bigcup_{b\in B} Y(k, [a, a], [b, b]).$$

In simulations, we partition the intervals A and B into n and m equal width subintervals, denoted as A_i and B_j for $i = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, m$, respectively. For each pair of i and j, we define the interval state equations and the interval output equation as:

$$\begin{aligned} X_{1ij}(k+1, A_i, B_j) &= [X_{1ij}^L(k+1, A_i, B_j), X_{1ij}^R(k+1, A_i, B_j)] \\ &= X_{2ij}(k, A_i, B_j), \\ X_{2ij}(k+1, A_i, B_j) &= [X_{2ij}^L(k+1, A_i, B_j), X_{2ij}^R(k+1, A_i, B_j)] \\ &= A_i \cdot X_{1ij}(k, A_i, B_j) + B_j \cdot X_{1ij}(k, A_i, B_j) + U(k) \end{aligned}$$

and

$$\begin{aligned} Y_{ij}(k, A_i, B_j) &= [Y_{ij}^L(k, A_i, B_j), Y_{ij}^R(k, A_i, B_j)] \\ &= X_{2ij}(k, A_i, B_j) \end{aligned}$$

for $k = 1, 2, \cdots$. By taking the union of $X_{1ij}(k, A_i, B_j)$, $X_{2ij}(k, A_i, B_j)$ and $Y_{ij}(k, A_i, B_j)$ for each pair (i, j), $1 \leq i \leq n$ and $1 \leq j \leq m$, we have

$$\begin{aligned} \tilde{X}_{1nm}(k, A, B) &= \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} X_{1ij}(k, A_i, B_j), \\ \tilde{X}_{2nm}(k, A, B) &= \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} X_{2ij}(k, A_i, B_j) \end{aligned}$$

and

$$\tilde{Y}_{nm}(k, A, B) = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} Y_{ij}(k, A_i, B_j).$$

On the other hand, if we are directly appling the interval arithmetic to this second digital filter, we have following results:

$$\begin{aligned} X_1(k+1,A,B) &= [X_1^L(k+1,A,B), X_1^R(k+1,A,B)] \\ &= X_2(k,A,B), \\ X_2(k+1,A,B) &= [X_2^L(k+1,A,B), X_2^R(k+1,A,B)] \\ &= A \cdot X_1(k,A,B) + B \cdot X_2(k,A,B) + U(k) \end{aligned}$$

and

$$\begin{aligned} Y(k,A,B) &= [Y^L(k,A,B), Y^R(k,A,B)] \\ &= X_2(k,A,B). \end{aligned}$$

From Theoreom 4.1, we would expect that

$$\begin{aligned} X_1(k,A,B) \supset \tilde{X}_{1nm}(k,A,B) \supset \lim_{n\to\infty} \lim_{m\to\infty} \tilde{X}_{1nm}(k,A,B) = \bar{X}_1(k,A,B), \\ X_2(k,A,B) \supset \tilde{X}_{2nm}(k,A,B) \supset \lim_{n\to\infty} \lim_{m\to\infty} \tilde{X}_{2nm}(k,A,B) = \bar{X}_2(k,A,B) \end{aligned}$$

and

$$Y(k,A,B) \supset \tilde{Y}_{nm}(k,A,B) \supset \lim_{n\to\infty} \lim_{m\to\infty} \tilde{Y}_{nm}(k,A,B) = \bar{Y}(k,A,B).$$

It can be shown that the digital filter of figure 1 has an eqilibrium $(X_1, X_2)^T = (0,0)$ which is asymptoticly stable in the large, provided that the parameter pair (a,b) is inside of the triangle shown in figure 2.

In addition to the simulations that were used to verify our theorem (utilizing interval analysis techniques), we also simulated the filter of figure 1 for 10^4 individual points in the $a - b$ parameter plane. These points were determined by the locations of the centers of the rectangles making up the grid of the parameter interval $\mathbf{I} = A \times B = ([a_1, a_2], [b_1, b_2])$. By abuse of notation, we refer to this collection of simulation runs as a “Monte Carlo-like method”.

Figures 4, and 5, present simulation runs where $a_1 = -0.6$, $a_2 = -0.4$, $b_1 = -0.3$, and $b_2 = -0.1$, i.e., $\mathbf{I} = A \times B = [-0.6, -0.4] \times [-0.3, -0.1]$. Figure 4 demonstrates the effects of the number of subintervals on the degree of conservatism of the estimated results. The pair of plots marked with +'s indicates the upper and lower bounds of the estimated solution bound with a 1 × 1 subinterval, i.e., the results obtained by directly applying interval arithmetic without subdividing $\mathbf{I}$. The pair of plots marked with diamonds and o's indicates the upper and lower estimated solution bounds with partitions using 10 × 10 and 50 × 50 subintervals, respectively. We discovered that the estimated solution bound becomes sharper as the number of subintervals is increased. In figure 5, the pair of plots indicated by dashs indicate the results of 100 × 100 point runs, while the plots marked with o's and x's indicate the estimated upper and lower solution bounds with 50 × 50 subintervals, respectively. Figures 4 and 5 verify our theorem.

REFERENCES.

[1] R. E. Moore, “Interval Arithmetic and Automatic Error Analysis in Digital Computing,” Ph. D. dissertation, Stanford University, 1962.

[2] R. E. Moore, Interval Analysis, Prentice-Hall, Inc., 1966.

[3] E. P. Oppenheimer and A. N. Michel, “Application of Interval Analysis Techniques to Linear Systems : Part I-Fundamental Results,” *IEEE Trans. on Circuits and Systems*, Vol. 35, No. 9, pp. 1129-1138, September, 1988.

[4] E. P. Oppenheimer and A. N. Michel, "Application of Interval Analysis Techniques to Linear Systems : Part II-Exponential Function," *IEEE Trans. on Circuits and Systems*, Vol. 35, No. 10, pp. 1230-1242, October, 1988.
[5] E. P. Oppenheimer and A. N. Michel, "Application of Interval Analysis Techniques to Linear Systems : Part III-Initial-value problems," *IEEE Trans. on Circuits and Systems*, Vol. 35, No. 10, pp. 1243-1256, October, 1988.
[6] K. Okumura, A. Kishima and S. Saeki, "Improvement of Algorithm Using Interval Analysis for Solution of Nonlinear Circuit Equations," *Electronics and Communications in Japan*, Part 1, Vol. 70, No. 9, pp. 1-10, 1987.
[7] S. Skelboe, "True Worst-Case Analysis of Linear Electrical Circuits by Interval Arithmetic," *IEEE Trans. on Circuits and Systems*, Vol. CAS-26, pp. 874-879, October, 1979.
[8] M. R. Valenca, "Multiple Shooting Using Interval Analysis," BIT 25, pp. 425-427, 1985.
[9] Ch. Jansson, "Self-validating Method for Solving Linear Programming Problems with Interval Input Data," Scientific Computation with Automatic Result Verification, edited by U. Kulisch and H. J. Stetter, pp. 33-46, Springer-Verlag, Wien, New York, 1988.
[10] H. Behnke, "Inclusion of Eigenvalues of General Eigenvalue Problems of Matrices," Scientific Computation with Automatic Result Verification, edited by U. Kulisch and H. J. Stetter, pp. 69-78, Springer-Verlag, Wien, New York, 1988.
[11] A. N. Michel; C. J. Herget, Mathematical Foundations in Engineering and Science, Prentice-Hall, Inc., 1981.
[12] A. E. Taylor and D. C. Lay, Introduction to Functional Analysis, 2nd Edition, John Wiley & Sons, Inc. 1980.
[13] E. P. Oppenheimer, "Application of interval analysis to problems of linear control systems," Ph. D. Thesis, Iowa State University, 1974.
[14] A. W. Naylor and G. R. Sell, Linear Operator Theory in Engineering and Science, Springer-Verlag New York Inc., 1982.
[15] A. E. Taylor, General Theory of Functions and Integration, Blaisdell Publishing Company, 1965.
[16] R. H. Kasriel, Undergraduate Topology, W. B. Saunders Company, 1971.
[17] R. L. Wheeden and A. Zygmund, Measure and Integral, Marcel Dekker, Inc., 1977.

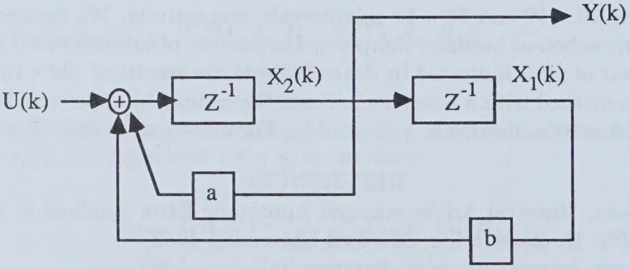


Figure 1.

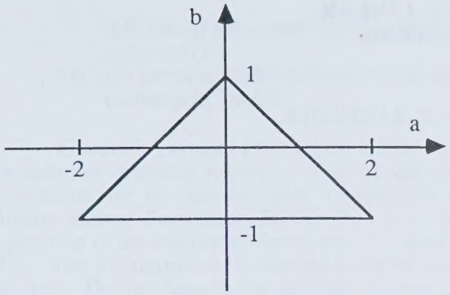


Figure 2.

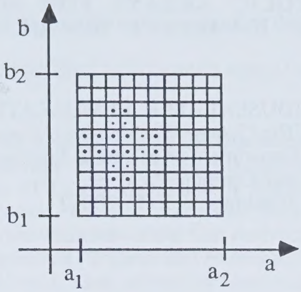

b
b_2
b_1
a_1
a_2
a

Figure 3.

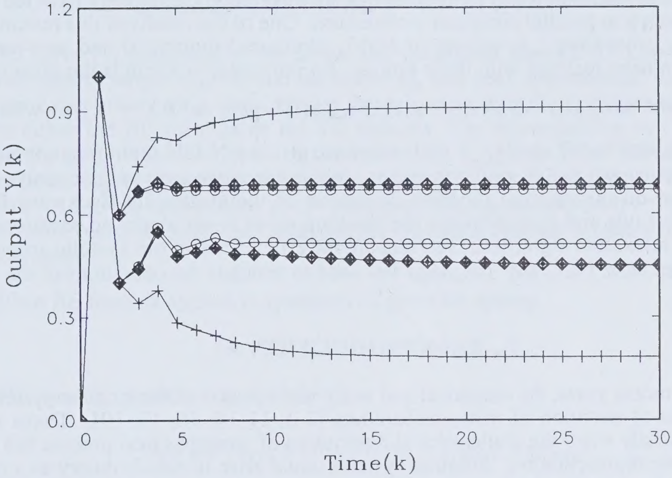


Figure 4.

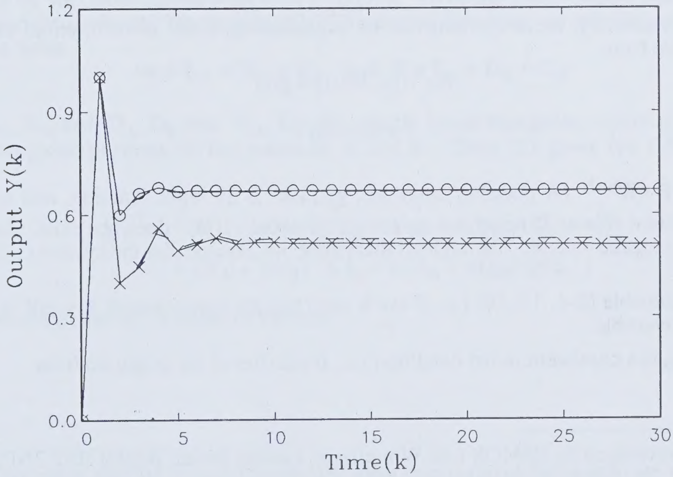


Figure 5.

SYSTOLIC ARRAYS FOR SOLVING LINEAR TIME INVARIANT SINGULAR SYSTEMS

ABDUSSALAM K. ELMUSRATI and M. E. ZAGHLOUL
The George Washington University
Department of Electrical Engineering
and Computer Science
Washington, D.C. 20052

ABSTRACT

Recent advances in **VLSI** (Very Large Scale Integration) technology have led to rapid growth in research in parallel computer architecture. One of the results of this research is the systolic array processors. A variety of highly structured numerical and non-numerical problems have been realized with these arrays. An important problem is the class of linear time invariant singular systems: $\mathbf{A}\mathbf{x}'(t) + \mathbf{B}\mathbf{x}(t) = \mathbf{g}(t)$; $\mathbf{x}(0) = \mathbf{x}_0$, where $\mathbf{A}$ (possibly singular) and $\mathbf{B}$ are $N \times N$ real matrices, $\mathbf{g}(t)$ is a N-fold input function, and $\mathbf{x}_0$ is a given N-dimensional initial condition vector. In this paper, we used systolic arrays to solve a model based on discretizing the first derivative of the system by Backward Euler or Trapezoidal formula and then decouple the resultant set of linear algebraic equations by the Gauss-Jacobi relaxation algorithm. To compare the efficiencies of our systolic arrays and to test their correctness, OCCAM[1] language was used to simulate the operations of the systolic arrays.

2. INTRODUCTION

In the recent years, the numerical and analytical solution of the singular systems have been the focus of attention of many researchers [1-3, 11-14, 16, 17, 19]. These systems arise very naturally when the mathematical description of some physical process fail to exist in state variable representation. Singular systems could arise in circuit theory as a result of combining the laws of circuit elements and the laws of interconnections [14]. These systems can also be the result of many other applications [1-3, 12, 20].

Mathematically, these systems can be expressed by a set of differential algebraic equations of the form

$$\mathbf{A}\mathbf{x}'(t) + \mathbf{B}\mathbf{x}(t) = \mathbf{g}(t) \tag{1.a}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{1.b}$$

where $\mathbf{A}$ and $\mathbf{B} \in \mathbb{R}^{N\times N}$ ($\mathbf{A}$ possibly singular), $\mathbf{g}(t) \in \mathbb{R}^N$ is an input function, and $\mathbf{x}_0 \in \mathbb{R}^N$ is an initial condition vector; $\mathbb{R}$ being the set of real numbers. If $\mathbf{A}^{-1}$ does not exist, then (1) is referred to as singular system. Throughout this paper, we assume that the system (1) is

(1) - solvable [2-4, 13, 16] i.e., $\exists\ \omega \in \mathbb{R}$ such that the matrix pencil $\mathbf{A} + \omega\mathbf{B}$ is invertible.

(2) - $\mathbf{x}_0$ is a consistent initial condition i.e., it satisfies at the origin both the

---

[1] OCCAM is developed by INMOS Ltd, Whitefriars, Lewins Mead, Bristol BS1 2NP, UK (1983).

differential sub-system and the algebraic constraints (if any) contained in the system (1).

(3) - the iterative model (which will be derived and realized with systolic arrays) is a convergent model.

Generally, system (1) tends to be stiff[2], requiring a stiffly stable [5] discretization formula for stability reasons. Two of the most common stiffly stable discretization algorithms are the second-order Trapezoidal formula (TR) and the class of Backward Differentiation Formulae (BDFs) [17, 19]. In this work, we will study the hardware realization of the solution of the system (1) based on the discretization of the first derivative of (1.a). The BE algorithm (Backward Euler) or the TR algorithm (Trapezoidal Formula) could be used. The resultant linear algebraic system of equations is then solved by Gauss-Jacobi relaxation algorithm.

3. MATHEMATICAL FORMULATION

To solve (1) numerically, we divide the simulation interval $[0, Lh = t_{end}]$ into L subintervals each of length h, $t_i = ih$ and set $\mathbf{x}(\mathbf{0}) = \mathbf{x}_0$, and seek approximate values, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$ to the exact values $\mathbf{x}(t_1), \mathbf{x}(t_2), \mathbf{x}(t_3), \ldots$ by approximating the first derivative of x(t) at $t = t_r$ with either the BE formula or the TR formula. The discretization of (1) in the *rth* subinterval $[t_r, t_{r+1}]$; $r = 0, 1, \ldots, end\text{-}1$ of the interval $[0, t_{end}]$ with either BE or TR formula leads to a linear algebraic system of equations in the unknown vector $\mathbf{x}_{r+1}$. To obtain the solution at $t = t_{r+1}$ given the solution at $t = t_r$ (from preceding simulation if $r > 0$ or from the initial condition vector if $r = 0$), the Gauss-Jacobi relaxation scheme is used to decouple this algebraic system.

When BE formula applied to system (1) it gives the system

$$(\mathbf{A} + h\mathbf{B})\mathbf{x}_{r+1} = \mathbf{A}\mathbf{x}_r + h\mathbf{g}(t_{r+1}) \tag{2}$$

and the TR algorithm produces the system

$$(\mathbf{A} + \frac{h}{2}\mathbf{B})\mathbf{x}_{r+1} = (\mathbf{A} - \frac{h}{2}\mathbf{B})\mathbf{x}_r + \frac{h}{2}[\mathbf{g}(t_r) + \mathbf{g}(t_{r+1})] \tag{3}$$

These are algebraic systems of equations in the unknown vector $\mathbf{x}_{r+1}$ and can be solved iteratively by the Gauss-Jacobi relaxation algorithm. To obtain Gauss-Jacobi Backward Euler (GJBE) and Gauss-Jacobi Trapezoidal (GJTR) iteration formulae, we split the matrices **A** and **B** into the sums

$$\mathbf{A} = \mathbf{L_A} + \mathbf{D_A} + \mathbf{U_A} \quad \text{and} \quad \mathbf{B} = \mathbf{L_B} + \mathbf{D_B} + \mathbf{U_B}$$

where $\mathbf{L_A}, \mathbf{L_B}$ and $\mathbf{D_A}, \mathbf{D_B}$ and $\mathbf{U_A}, \mathbf{U_B}$ are strictly lower triangular, diagonal, and strictly upper triangular portions of the matrices **A** and **B**. Then (2) gives the GJBE iterative formula

$$\begin{aligned}\mathbf{x}_{r+1}^{(\tau+1)} = &- (\mathbf{D_A} + h\mathbf{D_B})^{-1}\{(\mathbf{L_A} + \mathbf{U_A}) + h(\mathbf{L_B} + \mathbf{U_B})\}\,\mathbf{x}_{r+1}^{(\tau)} \\ &+ (\mathbf{D_A} + h\mathbf{D_B})^{-1}\mathbf{A}\,\mathbf{x}_r + h(\mathbf{D_A} + h\mathbf{D_B})^{-1}\mathbf{g}(t_{r+1})\end{aligned} \tag{4}$$

and (3) results in the GJTR iterative formula

---

[2] By stiff we mean that the natural frequencies (time constants) of the system are widely separated.

$$\mathbf{x}_{r+1}^{(\tau+1)} = -\left(\mathbf{D}_A + \frac{h}{2}\mathbf{D}_B\right)^{-1}\{(\mathbf{L}_A + \mathbf{U}_A) + \frac{h}{2}(\mathbf{L}_B + \mathbf{U}_B)\}\, \mathbf{x}_{r+1}^{(\tau)} \tag{5}$$
$$+ \left(\mathbf{D}_A + \frac{h}{2}\mathbf{D}_B\right)^{-1} \left(\mathbf{A} - \frac{h}{2}\mathbf{B}\right) \mathbf{x}_r + \frac{h}{2}\left(\mathbf{D}_A + \frac{h}{2}\mathbf{D}_B\right)^{-1}[\mathbf{g}(t_r) + \mathbf{g}(t_{r+1})]$$

where the superscript $\tau = 0, 1, 2, \ldots$ is the iteration count and and the subscript $r = 0, 1, 2, \ldots,$ *end*-1 is the time index. These iterative formulae have the same form and they can be encompassed in the model

$$\mathbf{x}_{r+1}^{(\tau+1)} = \mathrm{A0}\, \mathbf{x}_{r+1}^{(\tau)} + \mathrm{A1}\, \mathbf{x}_r + \mathbf{u}(t_r, t_{r+1}) \tag{6.a}$$

$$\mathbf{x}_0^{(\tau)} = \mathbf{x}_0 \tag{6.b}$$

Table-1 gives the GJBE and GJTR model parameters in terms of the original system parameters **A**, **B**, and **g**(t).

This model is an explicit one-step iterative formula since $\mathbf{x}_{r+1}^{(\tau+1)}$ depends on $\mathbf{x}_r$ and $\mathbf{x}_{r+1}^{(\tau)}$ but not on the past values $\mathbf{x}_{r-1}, \mathbf{x}_{r-2}, \mathbf{x}_{r-3}, \ldots$. Its convergence (and its speed of convergence) will essentially depend on the spectral properties of the iteration matrix A0.

Convergence is guaranteed only when $\max_{1 \leq i \leq N} |\lambda_i(\mathrm{A0})| < 1$, where $\lambda_i(\mathrm{A0})$ is the *i th* eigenvalue of the matrix A0. This is a well known result and its proof can be found in most numerical analysis books [15]. Letting $\mathbf{z}_{r+1}^{(\tau+1)} = \mathrm{A0}\mathbf{x}_{r+1}^{(\tau)}$ and $\mathbf{y}_r = \mathrm{A1}\mathbf{x}_r + \mathbf{u}(t_r, t_{r+1})$, rewrites (6.a) as

$$\mathbf{x}_{r+1}^{(\tau+1)} = \mathbf{z}_{r+1}^{(\tau+1)} + \mathbf{y}_r \tag{7}$$

The computation burdens of (7) are the computations of the two vectors $\mathbf{z}_{r+1}^{(\tau+1)}$and $\mathbf{y}_r$ and any attempt to speed-up its computations should be aimed to the computations of these vectors.

4. SYSTOLIC ARRAYS FOR A FIXED ORDER MODEL

The recent advent of **VLSI** technology has introduced a new architectural horizon in implementing parallel algorithms directly in hardware [6-11, 18]. This possibility has created a new research front on **VLSI** structures, as reported in [6-11, 18]. Kung and Leiserson [7] have introduced the notion of systolic arrays. In systolic arrays concept, **VLSI** devices consists of arrays of interconnected primitive processing elements PEs with distributed control and high degree of modularity. Each PE operates on strings of data that flow regularly and rhythmically through the array and is capable of performing a small set of operations and has some local memory and control logic. Strictly next-neighbor type connection constitute the interprocess communication structure and data moves through the architecture in a synchronous manner. This local interconnection efficiently utilize area, time, and energy of the silicon. Communication with the outside world takes place only at the array boundaries. Kung [7] gave some design criteria for designing systolic arrays. A variety of highly structured compute-bound problems have been realized with these arrays.

The iterative nature of (7) derives from the fact that the simulation procedure repeats when the time parameter t_r is advanced to t_{r+1} and L simulations are needed to realize the model computations on an interval of the form $[0, Lh = t_{end}]$. On each subinterval $[t_r, t_{r+1}]$, the model is a fixed point iteration algorithm whose fixed point $\mathbf{x}_{r+1}$ given the solution vector $\mathbf{x}_r$

and the input function $\mathbf{u}(t_r, t_{r+1})$. On the r*th* subinterval $[t_r, t_{r+1}]$, the vector $\mathbf{z}_{r+1}^{(\tau+1)}$ is updated several times (depending on the time step h, convergence properties of the model, and on the iteration stopping criterion), whereas the vector $\mathbf{y}_r$ is computed only once.

For the sake of illustration, we will confine ourselves to order (N = 3) model and to the computations of the $(\tau+1)th$ iteration in the r*th* subinterval $[t_r, t_{r+1}]$. The data dependency of the matrix-vector multiplication $\mathbf{z}_{r+1}^{(\tau+1)} = \mathbf{A0x}_{r+1}^{(\tau)}$ is shown by the data dependency graph (DDG) of Fig. 1. $PE_{i,j}$'s local memory is preloaded with the matrix element $a0_{i,j}$ and configured to implement at time i + j - 1 the single assignment relation $z_{i,r+1}^{(\tau+1)} = z_{i,r+1}^{(\tau+1)} + a0_{i,j} * x_{j,r+1}^{(\tau)}$. The computing time of order N matrix-vector multiplication on a DDG with N^2 PEs is $T = max\{|i+j-1| : i, j = 1, 2, ..., N\} = 2N - 1$ time steps[3] which is the longest path on the DDG and it is also the time required to produce the last component $z_{N,r+1}^{(\tau+1)}$.

The DDG of Fig. 1 computes the matrix-vector multiplication in 2N - 1 time steps as opposite to $2N^2 - N$ (N^2 multiplications and $N^2 - N$ additions) operations taken by a uni-processor machine. This results in O(N) speed-up, but inefficient utilization of the PEs since each PE can be active (or doing useful work) only for a small fraction of the life time of the array. In order to improve the PEs utilization[4] and hence the array utilization, it is usually desirable to map the two dimensional grid of processing elements onto arrays with fewer number of PEs and have regular topology. To accomplish this, we project the DDG of Fig. 1 along projection directions to yield linear arrays which preserve the computation time and has regular topology. A natural projection directions for a square grid is to project the grid along directions parallel to the coordinate axes or along directions parallel to its main diagonal. The coordinate direction projections give linear arrays with N PEs whose graphs are topologically equivalent to the graph of Fig. 2, where PE_i (i = 1, 2, ..., N) is the projection of all $PE_{i,j}$ (j = 1, 2, ..., N). The diagonal projection direction yield linear array with 2N - 1 PEs whose graph is shown in Fig. 3. PE_k $(-(N-1) \leq k \leq (N-1))$ of Fig. 3 is the projection of all $PE_{i,j}$ of Fig. 1 with k = i - j. In Figs. 2 and 3, a circle on a loop represents a unit delay. Figs. 4 and 5 depict respectively the systolic arrays of Figs. 2 and 3 and their PEs' operations after proper data skewing. The systolic arrays of Figs. 4a and 5 are pure systolic arrays (no data broadcast[5] is permitted), while the systolic array of Fig. 4b allows data to be broadcasted by the host to all the PEs.

To realize the computations of (7) on $[0, t_{end}]$ with these systolic arrays, we need to find the model solution at $t = t_{r+1}$ given its solution at $t = t_r$. This leads to *end* problem instances on the intervals $[0, t_1]$, $[t_1, t_2]$, ..., and $[t_{end-1}, t_{end}]$ and for single-step algorithm the r*th* problem instance on $[t_r, t_{r+1}]$ requires the solution of the (r -1)*th* problem instance on $[t_{r-1}, t_r]$ at $t = t_r$ and hence the global dependency among problem instances r and r - 1. This results in total ordering (chain dependency) of the subintervals simulations on the interval $[0, t_{end}]$.

Thus to solve, we can use one of our systolic arrays of fig. 4 or 5 to perform both matrix-vector multiplications (FIG. 6) in a serial-parallel pipelined manner. To simulate the computations of (7) on the r*th* subinterval $[t_r, t_{r+1}]$, we first use the systolic array to perform the computation $\mathbf{y}_r = \mathbf{A1x}_r + \mathbf{u}(t_r, t_{r+1})$. This can be accomplished by preloading (for Fig. 4) the local memory of each PE_i with i*th* input component $u_i(t_r, t_{r+1})$ which serves as initial value for $y_{i,r}$. In a pipeline fashion, the elements of the matrix **A1** are then fed from the north ports to the systolic array and the components of the vector $\mathbf{x}_r$ enter the array from its left. Then sequentially and repeatedly (until some stopping criterion is met), the computations of $z_{r+1}^{(\tau+1)}$ are pipelined to the systolic array (Fig. 6). Prior to the first iteration, the vector component $z_{j,r+1}^{(\tau+1)}$ is initialized with $y_{j,r}$.

---

[3] Time step is the time takes a PE to get its data and perform its operation(s).

[4] PEs utilization is the persent of time when the PEs performing computations.

[5] Data broadcasting is the process of sending a data to all processing elements at the same time in a multiprocessor system.

5. SYSTOLIC ARRAYS FOR UNRESTRICTED ORDER MODELS

To realize the computations of order N model by systolic arrays with n PEs, where $N \gg n$, we partition the model parameters A0 and A1 (the other parameters are partitioned accordingly) into stripes of dimension $n \times N$ or $N \times n$ then these stripes are piled along the their short sides to form a long stripe which is fed to the systolic array. Implicitly, we have assumed that n divides N. If this is not the case, the the system (1) is augmented with $n\lceil N/n\rceil$[6]- N first order ODEs $x_j'(t) + x_j(t) = 0$; $x_j(0) = 1$ for $j = N+1, \ldots, n\lceil N/n\rceil$ or extend the model (6) parameters with components which reflect the introduction of these ODEs (Table-2). Two partitioning algorithms will be used, horizontal and vertical partitioning. To illustrate the partitioning of matrix-vector multiplication computations, consider the partitioning of the computations $\mathbf{y}_r = \mathbf{A1}\mathbf{x}_r + \mathbf{u}(t_r, t_{r+1})$ to subproblems which fit the systolic arrays of Fig. 4. For N = 6 and n = 3, this matrix-vector multiplication can be expressed in component form as

$$\begin{pmatrix} y_{1,r} \\ y_{2,r} \\ y_{3,r} \\ y_{4,r} \\ y_{5,r} \\ y_{6,r} \end{pmatrix} = \begin{pmatrix} a1_{1,1} & a1_{1,2} & a1_{1,3} & a1_{1,4} & a1_{1,5} & a1_{1,6} \\ a1_{2,1} & a1_{2,2} & a1_{2,3} & a1_{2,4} & a1_{2,5} & a1_{2,6} \\ a1_{3,1} & a1_{3,2} & a1_{3,3} & a1_{3,4} & a1_{3,5} & a1_{3,6} \\ a1_{4,1} & a1_{4,2} & a1_{4,3} & a1_{4,4} & a1_{4,5} & a1_{4,6} \\ a1_{5,1} & a1_{5,2} & a1_{5,3} & a1_{5,4} & a1_{5,5} & a1_{5,6} \\ a1_{6,1} & a1_{6,2} & a1_{6,3} & a1_{6,4} & a1_{6,5} & a1_{6,6} \end{pmatrix} \begin{pmatrix} x_{1,r} \\ x_{2,r} \\ x_{3,r} \\ x_{4,r} \\ x_{5,r} \\ x_{6,r} \end{pmatrix} + \begin{pmatrix} u_1(t_r, t_{r+1}) \\ u_2(t_r, t_{r+1}) \\ u_3(t_r, t_{r+1}) \\ u_4(t_r, t_{r+1}) \\ u_5(t_r, t_{r+1}) \\ u_6(t_r, t_{r+1}) \end{pmatrix} \quad (8)$$

The vertical partitioning algorithm applied to (8) can be written in a block form as

$$\mathbf{y}_r = \mathbf{A1}_{1,1}\mathbf{x}_{1,r} + \mathbf{A1}_{1,2}\mathbf{x}_{2,r} + \mathbf{u}(t_r, t_{r+1})$$

where

$$\mathbf{x}_{1,r} = (x_{j,r}),\ j = 1, 2, 3;\ \ \mathbf{x}_{2,r} = (x_{j,r}),\ j = 4, 5, 6$$
$$\mathbf{A1}_{1,1} = (a1_{i,j}),\ j = 1, 2, 3,\quad \mathbf{A1}_{1,2} = (a1_{i,j}),\ j = 4, 5, 6\ ;\ i = 1, 2, \ldots, 6$$

The vector $\mathbf{y}_r$ can be computed by first feeding the sub-matrix $\mathbf{A1}_{1,1}$ to the systolic array from its north ports and loading the components of the vector $\mathbf{x}_{1,r}$ into its PEs with the *ith* component $(\mathbf{x}_{1,r})_i$ in the local memory of PE_i. The components of $\mathbf{y}_r$ (which are initialized with the corresponding components of the vector $\mathbf{u}(t_r, t_{r+1})$) enter the systolic array at its left end to leave its right end with partial values. This process computes the intermediate vector $\mathbf{y}_r^{temp} = \mathbf{A1}_{1,1}\mathbf{x}_{1,r} + \mathbf{u}(t_r, t_{r+1})$. To obtain the final vector $\mathbf{y}_r$, we pile the elements of the sub-matrix $\mathbf{A1}_{1,2}$ on top of the elements of the sub-matrix $\mathbf{A1}_{1,1}$ and load PE_i with the component $(\mathbf{x}_{2,r})_{n+i}$, then from the left end of the array we feed $\mathbf{y}_r^{temp}$ to collect the remaining inner product terms and leave the array from its right end with the final value $\mathbf{y}_r = \mathbf{y}_r^{temp} + \mathbf{A1}_{1,2}\mathbf{x}_{2,r}$.

For the horizontal partitioning algorithm, (8) can be expressed in the block form as

---

[6] $\lceil x\rceil$ is the smallest integer greater than or equal to x.

$$\begin{aligned} y_{1,r} &= A1_{1,1}\, x_r + u_1(t_r, t_{r+1}) \\ y_{2,r} &= A1_{2,1}\, x_r + u_2(t_r, t_{r+1}) \end{aligned}$$

where

$$y_{1,r} = (y_{i,r}),\ i = 1, 2, 3;\quad y_{2,r} = (y_{i,r}),\ i = 4, 5, 6$$
$$u_1(t_r, t_{r+1}) = (u_i(t_r, t_{r+1})),\ i = 1, 2, 3;\quad u_2(t_r, t_{r+1}) = (u_i(t_r, t_{r+1})),\ i = 4, 5, 6$$
$$A1_{1,1} = (a1_{i,j}),\ i = 1, 2, 3;\quad A1_{2,1} = (a1_{i,j}),\ i = 4, 5, 6;\ j = 1, 2, \ldots, 6$$

To compute $y_{1,r}$, PE$_i$'s local memory is preloaded with $u_i(t_r, t_{r+1})$. The components of the vector x_r move from the left end of the systolic array to its right end in a pipelined manner, and the elements of the block $A1_{1,1}$ are fed to the array from the north ports. At PE$_i$, $y_{j,r}$ get updated to $y_{j,r} + a1_{i,j} * x_{j,r}$ as it meets the matrix element $a1_{i,j}$ and the vector component $x_{j,r}$. In a similar fashion, after the final values of $y_{1,r}$ have been unloaded from the array the computation $y_{2,r} = A1_{2,1}x_r + u_2(t_r, t_{r+1})$ can be performed by loading the local memory of PE$_i$ with the *ith* component of the input vector $u_2(t_r, t_{r+1})$ which will be the initial value of $y_{2,r}$. The horizontal partitioning algorithm is more efficient than the vertical partitioning algorithm, since the sub-vectors $(y_{j,r})$, $j = 1,2$ are loaded to the systolic array only once while the vertical partitioning algorithm require two loadings these sub-vectors to the PEs and in general if $m = N/n$, m loadings are needed.

6. CONCLUSION

The systolic architectures presented here, were simulated and analyzed using the complexity measure (AT^2), where A denotes the array area which is measured by the number of its PEs, and T is the total (execution and communication) computing time. The programming language OCCAM have been used to simulate the operations of the systolic arrays and evaluate their performance based on this measure. The broadcast systolic array was found to be AT^2 optimum and with O(n) speed-up.

7. REFERENCES

[1] K. E. Bernan,"Solvability and Convergence for Higher Index Differential-Algebraic Systems with Application in Trajectory Control," Ph.D Thesis, UCLA, 1983.

[2] S. L. Campbell,"Singular Systems of Differential Equations," Research Notes in Mathematics, vol. 40, Pitman Publishing Co., Marshfield, Mass., 1979.

[3] __________,"Singular Systems of Differential Equations II," Research Notes in Mathematics, vol. 61, Pitman Publishing Co., Marshfield, Mass., 1982.

[4] __________,"Index Two Time Varying Singular Systems of Differential Equations,"SIAM J. Alg. Disc. Meth., 4(1983), no. 2, pp. 237-243.

[5] C. W. Gear,"The automatic integration of stiff ordinary differential equations," A. J. H. Morell,ed., in Information Processing 68, pp.187-193, Amsterdam,The Netherlands:North Holland, 1969.

[6] H. T. Kung,"Let's Design Algorithms for VLSI Systems," Proc. Caltech Conf. on Very Large Scale Integration, Jan. 1979, pp. 55-90.

[7] _______, and C. E. Leiserson,"Systolic Arrays for VLSI," in C. A. Mead and L. A. Conway, Introduction to VLSI systems. Reading Mass.: Addison Wesely, 1980, Section 8.3.

[8] _______,"Special-Purpose Devices for Signal Processing: An Opportunity in Very Large Scale Integration(VLSI)," Proc. SPIE vol. 242 Real-Time Signal Processing III, July 1980, pp. 76-84.

[9] _______, L. M. Ruance and D. D. I. Yen,"A Two-level Pipelined Systolic Arrays for Convolution in VLSI Systems and Computations", H. T. Kung et al. eds., Computer Science Press, Oct. 1981, pp. 255-264.

[10] _______,"Why Systolic Architecture?," IEEE Computer, vol. 15, no.1, Jan. 1982, pp. 37-46.

[11] E. Lelarsamee, A. E. Ruehli and, A. L. Sangiovanni-Vincentelli,"The Waveform Relaxation Method for Time Domain Analysis of Large Scale Integrated Circuits," University of California, Berkely, Electronics Research Laboratory, Memorandum no. UCB/ERL M81/75, June 1981.

[12] P. Lotstedt and L. R. Petzold,"Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints," Sandia Report SAND83-8877, Nov. 1983.

[13] D. G. Luenberger,"Dynamic Equations in Descriptor form," IEEE Trans. Aut. Cont.,vol. AC-22, pp. 312-321, 1977.

[14] R. W. Newcomb,"The Semistate Descriptor of Nonlinear Time-variant Circuits," IEEE Circuits Sys., vol. GAS-28, no. 1, pp. 62-71, Jan. 1981.

[15] J. M. Ortega and W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, 1970.

[16] L. R. Petzold,"Differential/Algebraic Equations are not ODE's," SIAM J. Sci. Stat. Comput., 3(1982), no. 3, pp. 367-384.

[17] __________, and C. W. Gear, "ODE Methods for the solution of Differential/Algebraic Systems," Sandia Report SAND82-8051, Oct. 1982.

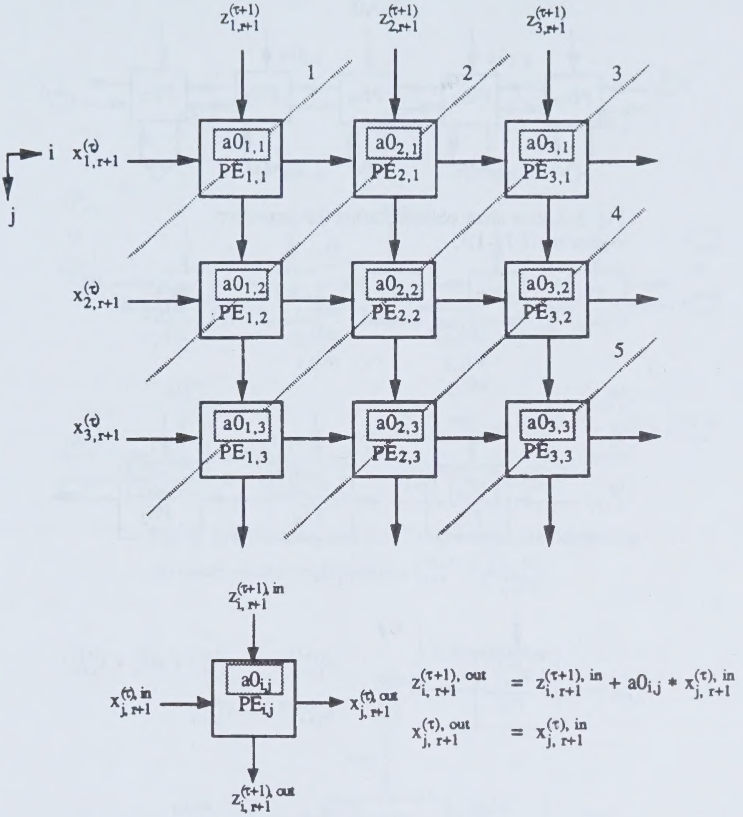
[18] P. Quinton,"The Systematic Design of Systolic Arrays," IRISA Research Report no.193, April 1983.

[19] R. F. Sincovec, B. Dembart, M. Epton, A. M. Erisman, J. W. Manke, and E. L.Yip "Solvability of Large Scale Descriptor Systems," Final Report, Boeing Computer Company, Tukwila, WA, June 1979.

[20] M. W. Spong,"A Semistate Approach to Feedback Stabilization of Neutral Delay Systems,"Circuit Sys. Sig. Proc., vol. 5, no. 1,pp. 69-85, 1986.

Model (6) parameters	GJBE parameters with $\mathbf{M}_{BE} = (\mathbf{D}_A + h\mathbf{D}_B)^{-1}$	GJTR parameters with $\mathbf{M}_{TR} = \left(\mathbf{D}_A + \frac{h}{2}\mathbf{D}_B\right)^{-1}$
A0	$-\mathbf{M}_{BE}^{-1}\{(\mathbf{L}_A + \mathbf{U}_A) + h(\mathbf{L}_B + \mathbf{U}_B)\}$	$-\mathbf{M}_{TR}^{-1}\{(\mathbf{L}_A + \mathbf{U}_A) + \frac{h}{2}(\mathbf{L}_B + \mathbf{U}_B)\}$
A1	$\mathbf{M}_{BE}^{-1}\mathbf{A}$	$\mathbf{M}_{TR}^{-1}(\mathbf{A} - \frac{h}{2}\mathbf{B})$
$\mathbf{u}(t_r, t_{r+1})$	$h\mathbf{M}_{BE}^{-1}\mathbf{g}(t_{r+1})$	$\frac{h}{2}\mathbf{M}_{TR}^{-1}[\mathbf{g}(t_r) + \mathbf{g}(t_{r+1})]$

Table-1: GJBE and GJTR model (6) parameters.

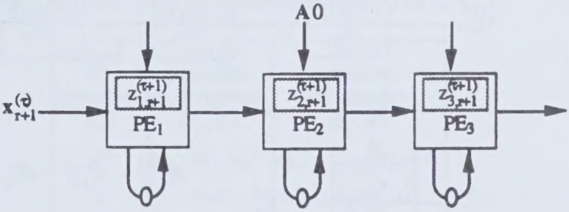


$z^{(\tau+1)}_{1,r+1}$
$z^{(\tau+1)}_{2,r+1}$
$z^{(\tau+1)}_{3,r+1}$
1
2
3
4
5
i
j
$x^{(\tau)}_{1,r+1}$
$x^{(\tau)}_{2,r+1}$
$x^{(\tau)}_{3,r+1}$
$a0_{1,1}$ $PE_{1,1}$
$a0_{2,1}$ $PE_{2,1}$
$a0_{3,1}$ $PE_{3,1}$
$a0_{1,2}$ $PE_{1,2}$
$a0_{2,2}$ $PE_{2,2}$
$a0_{3,2}$ $PE_{3,2}$
$a0_{1,3}$ $PE_{1,3}$
$a0_{2,3}$ $PE_{2,3}$
$a0_{3,3}$ $PE_{3,3}$
$z^{(\tau+1),\,in}_{i,\,r+1}$
$x^{(\tau),\,in}_{j,\,r+1}$
$a0_{i,j}$ $PE_{i,j}$
$x^{(\tau),\,out}_{j,\,r+1}$
$z^{(\tau+1),\,out}_{i,\,r+1}$

$$z^{(\tau+1),\,out}_{i,\,r+1} = z^{(\tau+1),\,in}_{i,\,r+1} + a0_{i,j} * x^{(\tau),\,in}_{j,\,r+1}$$

$$x^{(\tau),\,out}_{j,\,r+1} = x^{(\tau),\,in}_{j,\,r+1}$$

Fig. 1: DDG for third order matrix-vector multiplication $\mathbf{z}^{(\tau+1)}_{r+1} = A0\mathbf{x}^{(\tau)}_{r+1}$. PEs lie the hyper-plane κ are active at time κ.



A0
$x^{(\tau)}_{r+1}$
$z^{(\tau+1)}_{1,r+1}$
PE_1
$z^{(\tau+1)}_{2,r+1}$
PE_2
$z^{(\tau+1)}_{3,r+1}$
PE_3

Fig. 2: Linear array obtained by using the projection vector $\mathbf{p} = (0, 1)^T$.

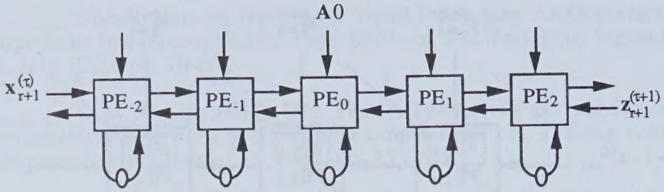
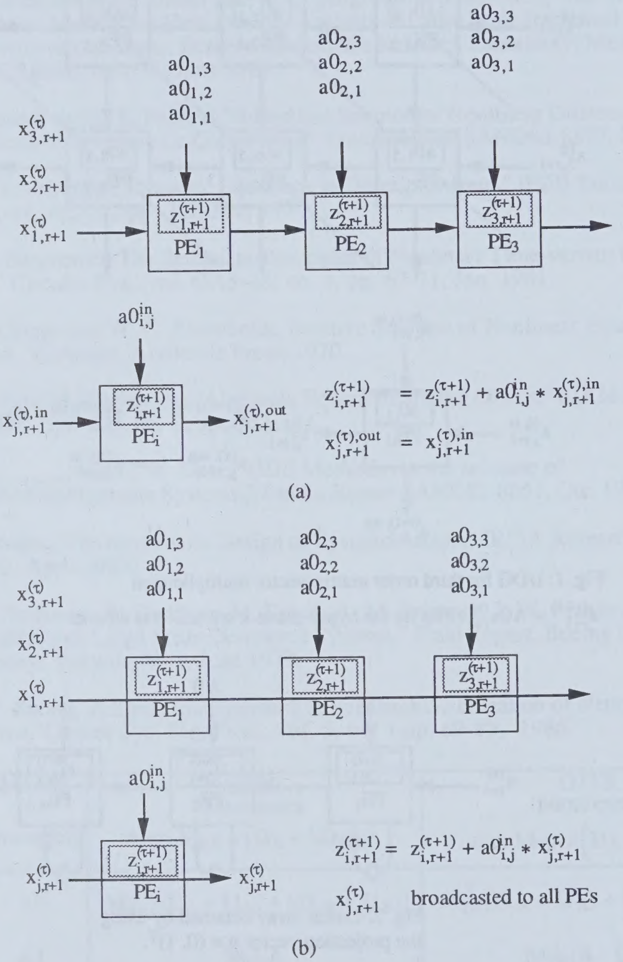

A0
x(τ) r+1
PE-2
PE-1
PE0
PE1
PE2
z(τ+1) r+1

Fig. 3: Linear array obtained using the projection vector $\mathbf{p} = (-1, -1)^T$.


a0 1,3
a0 1,2
a0 1,1
a0 2,3
a0 2,2
a0 2,1
a0 3,3
a0 3,2
a0 3,1
x(τ) 3,r+1
x(τ) 2,r+1
x(τ) 1,r+1
z(τ+1) 1,r+1
z(τ+1) 2,r+1
z(τ+1) 3,r+1
PE1
PE2
PE3
a0in i,j
x(τ),in j,r+1
z(τ+1) i,r+1
PEi
x(τ),out j,r+1
(a)
x(τ) j,r+1
broadcasted to all PEs
(b)

$$z^{(\tau+1)}_{i,r+1} = z^{(\tau+1)}_{i,r+1} + a0^{in}_{i,j} * x^{(\tau),in}_{j,r+1}$$

$$x^{(\tau),out}_{j,r+1} = x^{(\tau),in}_{j,r+1}$$

$$z^{(\tau+1)}_{i,r+1} = z^{(\tau+1)}_{i,r+1} + a0^{in}_{i,j} * x^{(\tau)}_{j,r+1}$$

Fig. 4: Systolic arrays and their PEs operations for computing the matrix-vector multiplication $z^{(\tau+1)}_{r+1} = A0x^{(\tau)}_{r+1}$.
(a) - Pure systolic array.
(b) - Data broadcast systolic array.

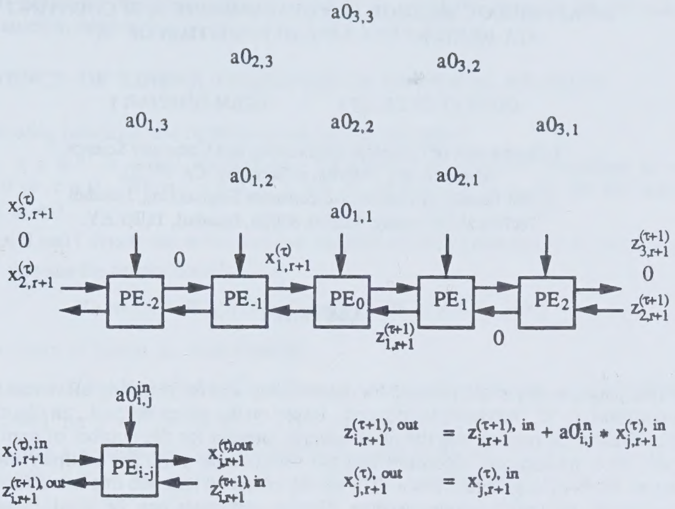

a0_{3,3}
a0_{2,3}
a0_{3,2}
a0_{1,3}
a0_{2,2}
a0_{3,1}
a0_{1,2}
a0_{2,1}
a0_{1,1}
x^{(τ)}_{3,r+1}
0
x^{(τ)}_{2,r+1}
0
x^{(τ)}_{1,r+1}
PE_{-2}
PE_{-1}
PE_0
PE_1
PE_2
z^{(τ+1)}_{3,r+1}
0
z^{(τ+1)}_{2,r+1}
z^{(τ+1)}_{1,r+1}
0
a0^{in}_{i,j}
x^{(τ),in}_{j,r+1}
PE_{i-j}
x^{(τ),out}_{j,r+1}
z^{(τ+1),out}_{i,r+1}
z^{(τ+1),in}_{i,r+1}
z^{(τ+1),out}_{i,r+1} = z^{(τ+1),in}_{i,r+1} + a0^{in}_{i,j} * x^{(τ),in}_{j,r+1}
x^{(τ),out}_{j,r+1} = x^{(τ),in}_{j,r+1}

Fig. 5: Systolic array and its PEs operations for computing the matrix-vector multiplication $z^{(\tau+1)}_{r+1} = A0x^{(\tau)}_{r+1}$.

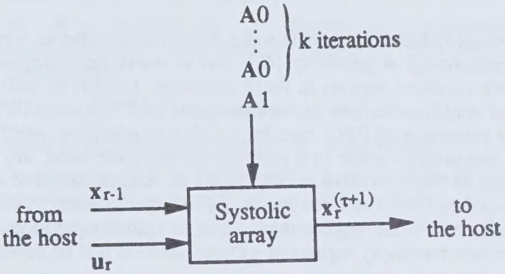

A0
⋮
A0
k iterations
A1
x_{r-1}
from the host
u_r
Systolic array
x^{(τ+1)}_r
to the host

Fig. 6: Block diagram for Pipelining k iterations on $[t_r, t_{r+1}]$.

Model (6) parameters	GJBE parameters augmentation	GJTR parameters augmentation
A0	$a0_{j,j}=0$	$a0_{j,j}=0$
A1	$a1_{j,j}=(1+h)^{-1}$	$a1_{j,j}=(1+\frac{h}{2})(1-\frac{h}{2})^{-1}$
u	$u_j=0$	$u_j=0$
x(0)	$x_j(0)=1$	$x_j(0)=1$

Table 2: Model (6) augmentation $j = N+1, \ldots, n\lceil N/n \rceil$

AN ALGEBRAIC METHOD FOR DETERMINING AND COUNTING ALL REGIONS IN A LINEAR PARTITION OF $\mathbf{R}^n$

CÜNEYT GÜZELIS † İ.CEM GÖKNAR ‡

† Departmant of Electrical Engineering and Computer Science,
University of California at Berkeley, CA 94720.
‡ The Faculty of Electrical-Electronics Engineering, Istanbul
Technical University. Maslak 80626, Istanbul, TURKEY.

ABSTRACT

In this paper,an algebraic method for determining and for counting all nonempty regions in a linear partition of $\mathbf{R}^n$ has been introduced. Based on the given method , an algorithm which is capable of finding all regions and the most general formula for the number of regions have been developed. The method, the algorithm and the formula are valid for arbitrary linear partitions including all kinds of degenerate cases. The results obtained here are important for finding multiple solutions of piecewise-affine resistive circuits and they can be used in other areas of piecewise-affine analysis of nonlinear circuits and systems as well as in computational geometry.

1. INTRODUCTION

Linear partitions induced by finitely many hyperplanes arise in connection with the problems subject to computational geometry [6,8] and in many optimization problems where constraint domains are specified regions in these partitions [12-13] as well as in piecewise-affine (PWA) analysis of nonlinear circuits as domain spaces of PWA maps [9,11]. It has been shown in [11] that many properties of PWA circuits are closely related to whether degenerate cases of parallelicity and concurrency occur in a partition.On the other hand, any combinatorial method finding all solutions of PWA resistive circuits needs the determination of all nonempty regions in a partition as a first step [10]. For these needs and in order to solve an old problem of combinatorial geometry , that how many regions there are in an arbitrary partition, an algebraic method to determine and to count nonempty regions of a linear partition will be developed in this paper.

The problem of determining the nonempty regions into which $\mathbf{R}^n$ is partitioned by a finite number of hyperplanes has been studied by many authors [1-7]. Since there are some nondegeneracy assumptions such as no parallel hyperplanes being allowed and/or more than two lines not intersecting at one point etc...,their approaches lack generality. Two recent computational methods can be identified in this area:sweep-plane method [1] and incremental construction [6,8]. In [1], an algorithm determining all regions is presented without lack of generality. However, all these methods are geometrical and a formula for the number of regions without restrictions has not been derived yet.

The method given here is purely algebraic and there is no restriction about the kind of partition of space and its dimension. This method is an application of convex set theory, it uses and generalizes the well known theorems of mathematical programming theory for solving linear partition problems.

This paper is structured as follows. In section 2 , as an application of the separating hyperplane theorem [12], the examination of consistency of linear inequalities characterizing regions and hence the determination of whether a region is empty or not have been transformed to finding the orhants having a basic solution which satisfies the angle condition. While an algorithm which determines all nonempty regions is given in section 3, a general formula for the number of regions is obtained in section 4.

2. CONSISTENCY OF LINEAR INEQUALITIES DEFINING REGIONS

The following notations and definitions are used in this paper.

i). A point $x \in \mathbf{R}^n$ is called nonnegative, semipositive or positive according to whether $x \geqq 0$, $x \geq 0$ or $x > 0$. Where, $x > 0$ if $x_i > 0$ for all i, $x \geqq 0$ if $x_i \geq 0$ for all i and $x \geq 0$ if $x \geq 0$ with $x \neq 0$.

ii). The letters S and I denote the set of positive integers which are subsets of $L := \{ 1,2,...,l \}$.

iii). $\mathbf{R}_\mathrm{I} \subseteq \mathbf{R}^l$ denotes the orhant described by

$$\mathbf{R}_\mathrm{I} := \{ y \in \mathbf{R}^l \mid y_i \geq 0 \quad i \in I \ , y_i \leq 0 \quad i \notin I \ , y \neq 0 \}$$

iv). The sign vector of region B_I is defined by

$$sgn\ (B_I) := (sgn\ (\alpha_1^T x + \beta_1) , sgn\ (\alpha_2^T x + \beta_2) , \ldots , sgn\ (\alpha_l^T x + \beta_l)) ; \quad x \in Interior\ (B_I)$$

here $sgn\ (t) = +1$ for $t > 0$, $= -1$ for $t < 0$, $= 0$ for $t = 0$.

v). A nonnegative (or respectively semipositive) combination of $y_1 , y_2 ,..., y_p \in \mathbf{R}^n$ is a linear combination $\lambda_1 y_1 + \lambda_2 y_2 + \cdots \lambda_p y_p$ with $\lambda \geqq 0$ (respectively $\lambda \geq 0$).

vi). The matrix $A \in \mathbf{R}^{l \times n}$ and the vector $b \in \mathbf{R}^l$ will be used instead of the matrix A_L and the vector b_L which will be defined in (4).

vii). A solution $y \in \mathbf{R}^l$ of the equation $y^T A = 0$ is called a basic solution if there exists a set S such that $y_i = 0$ for $i \notin S$ and the linear relation between the row vectors a_i of A for $i \in S$ is unique (up to multiplication by a nonzero constant) .

viii). A partition of $\mathbf{R}^n$ induced by l distinct (n-1)-dimensional hyperplanes $H_1 , H_2 , \ldots , H_l$ is called as a linear partition and is denoted by $\Pi := \{ H_1 , H_2 , \ldots , H_l \}$. Where hyperplanes are described by

$$H_i := \{ x \in \mathbf{R}^n \mid \alpha_i^\mathrm{T} \mathrm{x} + \beta_\mathrm{i} = 0 \} \quad , \quad i \in \{ 1,2,...,l \} \tag{1}$$

here α_i is the normal vector to the i'th hyperplane and β_i is a constant.

ix). The dimension of a subset of $\mathbf{R}^n$ is defined to be the dimension of the smallest dimensional affine subspace containing it. If a subset of $\mathbf{R}^n$ is of dimension n, it will be called proper.

x). A set $B \subseteq \mathbf{R}^n$ is convex if $x_1 , x_2 \in B$ implies that $\lambda x_1 + (1 - \lambda) x_2 \in B$,for all $\lambda \in [0 , 1]$.

xi). $F \subseteq \mathbf{R}^n$ is a flat if $x_1 \in F$ and $x_2 \in F$ imply that $\lambda x_1 + (1 - \lambda) x_2 \in F$ for all $\lambda \in \mathbf{R}$. And, a flat of dimension k will be called a k-flat.

xii). PWA maps are defined as maps whose domain spaces are linear partitions of $\mathbf{R}^n$ where the maps are affine in each region.

xiii). The symbol $\#B$ denotes the cardinality of the set B. ☐

Associated with a linear partition for each subset I of L there correspondens a different convex region B_I defined by

$$B_I := \{ x \in \mathbf{R}^n \mid \alpha_\mathrm{i}^\mathrm{T} \mathrm{x} + \beta_\mathrm{i} \geq 0 \quad \mathrm{i} \in \mathrm{I} \ , \alpha_\mathrm{i}^\mathrm{T} \mathrm{x} + \beta_\mathrm{i} \leq 0 \quad \mathrm{i} \notin \mathrm{I} \ \} \tag{2}$$

If a region B_I is a proper subset of $\mathbf{R}^n$, it will be called non-empty. It is clear that B_I is not empty if and only if the inequalities

$$A_I \, x - b_I > 0 \tag{3}$$

have a solution ,where the matrices A_I , b_I are formed by taking the i'th row as α_i^T and $-\beta_i$ for $i \in I$ and $-\alpha_i^T$, β_i for $i \notin I$,respectively. This is denoted by

$$A_I := \begin{bmatrix} \sigma_1 \cdot \alpha_1^T \\ \sigma_2 \cdot \alpha_2^T \\ \cdot \\ \cdot \\ \cdot \\ \sigma_l \cdot \alpha_l^T \end{bmatrix} , \qquad b_I := \begin{bmatrix} -\sigma_1 \cdot \beta_1 \\ -\sigma_2 \cdot \beta_2 \\ \cdot \\ \cdot \\ \cdot \\ -\sigma_l \cdot \beta_l \end{bmatrix} \tag{4}$$

here $\sigma_i = 1$ for $i \in I$ and $\sigma_i = -1$ for $i \notin I$.

Theorem 1 gives a necessary and sufficient condition for the consistency of the sytem of inequalities in (3) ,i.e., for the existence of a solution to (3). By Theorem 1, the checking of that a region is empty or not is tranformed into the examination of whether the solutions of a set of linear homogeneous equations satisfy an angle condition. Theorem 2 together with Theorem 3 give a criterion for the existence of a solution to the homogeneous equation system in an orhant $\mathbf{R}_I$. The following lemma of the separating hyperplane will be used to prove Theorem 1.

Lemma [15] **:** Let B be a convex set containing no positive point. Then there is a separating hyperplane $c^T x = 0$, $x \in \mathbf{R}^n$ having a semipositive normal $c \geq 0$ such that the half-space $c^T x \leq 0$ contains B. □

Theorem 1 [11] **:** Given $E \in \mathbf{R}^{l \times n}$, $w \in \mathbf{R}^l$ there exists $x \in \mathbf{R}^n$ such that

$$E\,x > w \tag{5}$$

if and only if there exist no semipositive $y \in \mathbf{R}^l$ such that

$$y^T E = 0 \ , \ y^T w \geq 0 \qquad \square \tag{6}$$

The Theorem 1 is similar to the well-known theorem on duality in mathematical programming [12-14]. The only difference between their statements is in the terms of ≥ and > (however, this necessitates some nontrivial modifications in the proof). Applying Theorem 1 to the inequality (3), it can be seen immediately that a region B_I is not empty if and only if the set $\hat{Y}$ is empty (or equivalently Y_I is empty) where

$$Y_I := \{ y \in \mathbf{R}^l \mid y \in \mathrm{R_I} \ , \mathrm{y^T A} = 0 \ , \mathrm{y^T b} \geq 0 \} \ , \ \hat{Y} := \{ y \in \mathbf{R}^l \mid y \in \mathrm{R_L} \ , \mathrm{y^T A_I} = 0 \ , \mathrm{y^T b_I} \geq 0 \} \tag{7}$$

It follows from Theorem 1 and (7) that instead of looking for the consistency of the inequality system (3) defining the region B_I , one could try to obtain all solutions to the homogeneous equations in (6) and to check if the ones in the orhant $\mathbf{R}_I$ satisfies the angle condition (i.e. $y^T b \geq 0$). This result is not very useful, since there are, in general, infinitely many solutions to a homogeneous equation system. However, as can be seen from Theorem 2 and 3 , all solutions to a system of homogeneous equations can be written as nonnegative combinations of finitely many specific solutions (which are called basic solutions). Therefore, the above task of finding the solutions to the system of homogeneous equations in (6) under the constraints can be accomplished by considering basic solutions only.

Theorem 2 [11] **:** If the equation

$$y^T A = 0 \tag{8}$$

has a solution in the orhant $\mathbf{R}_I$, it has a basic solution in the orhant $\mathbf{R}_I$. □

Theorem 3 [11] **:** Every solution of (8) in the orhant $\mathbf{R}_I$ is a semipositive combination of basic solutions in the orhant $\mathbf{R}_I$. □

Theorems 2 and 3 are an extension of the basic solution Theorems in [13] ; using them it can be shown that either there is a basic solution in Y_I or the region B_I is non-empty. In order to reach this result, it is sufficient to see that if the scalar pruducts of b with all basic solutions y in $\mathbf{R}_I$ are nonnegative then so are the scalar products of b with all solutions y in $\mathbf{R}_I$ and vice versa.

3. AN ALGORITHM FOR FINDING ALL REGIONS

The problem of determining all regions is reduced by Theorems 1-3 to that of finding all basic solutions of the homogenuous equations $y^T A = 0$ and checking the angle condition $y^T b \geq 0$ for these basic solutions only.

The determination of basic solutions to a system of linear equations is of great importance in linear programming and in other areas of mathematical progamming theory such as the enumaration of all vertices of a polyhedron. Consequently, there are many algorithms for determining basic solutions.

The algorithm given below allows the use of Gauss-Jordan elimination for finding basic solutions. Moreover, the scalar product of b with any basic solution y is calculated without additional effort by incorporating b into the matrix subject to Gauss-Jordan elimination.

In the proposed algoritm, the set of basic solutions for $y^T A = 0$ is obtained by an iterative procedure. Once a basis for the left null-space of A is found, then all basic solutions are obtained from this basis by elementary row operations and only column permutations.

Note that for an appropriate permutation matrix P, any matrix $A \in \mathbf{R}^{l \times n}$ can be written as $P\ A = L\ U$ in standard L-U form [14] . Here the upper trapezoidal matrix U has non-zero entries in echelon form. Starting from $P\ A = L\ U$ or $L^{-1}\ P\ A = U$ with $r = Rank\ (A\ )$, as the last $l - r$ rows of U are zero , the last $l - r$ rows of $L^{-1}\ P$ form a basis for the left null space [14].

Algorithm :

Input of the algorithm is l vectors

$$[\ \alpha_1^T\ \ \beta_1\ ]\ ,[\ \alpha_2^T\ \ \beta_2\ ]\ ,\ldots,\ [\ \alpha_l^T\ \ \beta_l\ ]$$

which define hyperplanes $H_1\ ,H_2\ ,\ldots,\ H_l$ .Output of the algorithm is the number of regions and the set of all nonempty regions given by their sign-vectors.

Step 1 : Construct the augmented matrix M

$$M := [\ A\ \ \ I_l\ \ \ b\ ] \tag{9}$$

where $A \in \mathbf{R}^{l \times n}$, I_l is the $l \times l$ identity matrix and $b \in \mathbf{R}^l$.

Step 2 : To find a basis for the left-nullspace of A , operate on A to produce an upper trapezoidal matrix U.

$$L^{-1}\ P\ M := \hat{M} = [\ U\ \ \ L^{-1}P\ \ \ L^{-1}Pb\ ] \tag{10}$$

Step 3 : Define the matrix $\hat{N}$ as

$$\hat{N} := [\ N\ \ \ Nb\ ] \tag{11}$$

where N and Nb are obtained by deleting the first r rows of $L^{-1}\ P$ and $L^{-1}\ P\ b$, respectively.

The rows of N form a basis for the left-nullspace of A .

Step 4 : Every basic solution can be obtained from N by Gauss-Jordan elimination. For this; let the column permutation matrix P^k operate on N , row operations will be carried out on the matrix $N P^k$. The matrix N is reduced by elementary row operations at each step to the row-reduced echelon form.

$$E^k [N \quad Nb] \begin{bmatrix} P^k & 0 \\ 0 & 1 \end{bmatrix} = [U^k \quad \tilde{N}^k \quad E^k Nb] \tag{12}$$

where $U^k \in \mathbf{R}^{(l-r)\times(l-r)}$ and $P^k := [e^{k_1} e^{k_2} \cdots e^{k_{l-r+1}} e^{k_{l-r+2}} \cdots e^{k_l}]$ in which e^{k_i} is the unit vector

for $k \in \{ 1,2 , \ldots , \binom{l}{l-r} \}$, $i \in \{ 1,2 , \ldots , l \}$ and the sets $\{ k_1 , k_2 ,\ldots, k_{l-r} \}$,

$\{ k_{l-r+1} , k_{l-r+2} ,\ldots, k_l \}$ are complementary subsets of L.

Note 1 : It is clear that every row of $E^k N$ is not a basic solution but every basic solution is a row of $E^k N$ (up to multiplication by a nonzero constant). ☐

Step 5 : Let $(E^k N)_j$ and $(E^k N b)_j$ denote the j'th row of $E^k N$ and j'th element of $E^k N b$, respectively. If $(E^k N b)_j > 0$, check whether $(E^k N)_j$ is in the orhant $\mathbf{R}_{\mathrm{I}}$ or not. Otherwise, if $(- E^k N b)_j > 0$, then there exists a solution $(- E^k N)_j$ satisfying the angle condition in $\mathbf{R}_{\mathrm{I}}$. On the other hand, if $(E^k N b)_j = 0$, both of the solutions $(E^k N)_j$ and $(- E^k N)_j$ satisfy the angle condition.

Note 2 : Step 5 is applied to all rows of $E^k N$ since to find the basic solutions among the rows of $E^k N$ requires considerably more effort. ☐

The number of nonempty regions is equal to the number of $\mathbf{R}_{\mathrm{I}}$ which do not contain a basic solution satisfying the angle condition. Note that the sign-vector of a region is obtained by the sign-vector of the associated orhant $\mathbf{R}_{\mathrm{I}}$.

Example 1 : The partition of the plane induced by the following straight lines is to be determined.

$$H_1 = [\alpha_1^T \ \beta_1] = [1 \ -1 \ \ 1] \quad ; \quad H_2 = [\alpha_2^T \ \beta_2] = [1 \ -1 \ \ 2]$$
$$H_3 = [\alpha_3^T \ \beta_3] = [1 \ 1 \ \ -1] \quad ; \quad H_4 = [\alpha_4^T \ \beta_4] = [0 \ 1 \ \ -1]$$

where H_1 and H_2 are parallel and H_1 , H_3 and H_4 pass through a common point.Let the above procedure be applied to this problem.

$$(S1) : M = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & 1 & 0 & 0 & -2 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad ; \quad (S2) : \hat{M} = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 2 & -1 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0.5 & 0 & -0.5 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & -1 \end{bmatrix}$$

$$(S3) : \hat{N} = \begin{bmatrix} 0.5 & 0 & -0.5 & 1 & 0 \\ -1 & 1 & 0 & 0 & -1 \end{bmatrix} \quad ; \quad (S4) : [E^1 N P^1 \quad E^1 Nb] = \begin{bmatrix} 1 & 0 & -1 & 2 & 0 \\ 0 & 1 & -1 & 2 & -1 \end{bmatrix}$$

$$(S4):[\,E^2NP^2 \;\; E^2Nb\,]=\begin{bmatrix}1 & 0 & -1 & 0 & 1\\ 0 & 1 & -1 & -2 & 1\end{bmatrix};(S4):[\,E^3NP^3 \;\; E^3Nb\,]=\begin{bmatrix}1 & 0 & -1 & 0 & 1\\ 0 & 1 & 0.5 & -0.5 & -0.5\end{bmatrix}$$

$$(S4):[\,E^4NP^4 \;\; E^4Nb\,]=\begin{bmatrix}1 & 0 & -1 & 0 & -1\\ 0 & 1 & 0.5 & -0.5 & 0\end{bmatrix} \;;\; (S4):[\,E^5NP^5 \;\; E^5Nb\,]=\begin{bmatrix}1 & -2 & 0 & -1 & 1\\ 0 & 0 & 1 & -1 & 1\end{bmatrix}$$

$$(S4):[\,E^6NP^6 \;\; E^6Nb\,]=\begin{bmatrix}1 & 0 & -1 & -2 & 0\\ 0 & 1 & -1 & 0 & -1\end{bmatrix}$$

Hence basic solutions of

$$y^T A=[\,y_1 y_2 y_3 y_4\,]\begin{bmatrix}1 & -1\\ 1 & -1\\ 1 & 1\\ 0 & 1\end{bmatrix}=[\,0\,0\,]$$

are $y_1=[\,1\,-1\,0\,0\,]^T$, $y_2=[\,1\,0\,-1\,2\,]^T$, $y_3=[\,0\,-1\,1\,-2\,]^T$ and their opposites in sign.But only y_1 , y_2 , $-y_2$ and y_3 satisfy angle condition

$$y^T b=[\,y_1 y_2 y_3 y_4\,]\begin{bmatrix}-1\\ -2\\ 1\\ 1\end{bmatrix}\geq 0$$

Empty regions are determined by testing which $\mathbf{R}_I$ contains the solutions satisfying the angle condition. Seven empty regions are found as $B_{\{1,3\}}$, $B_{\{1,2,4\}}$, $B_{\{1,4\}}$, $B_{\{2,3\}}$, $B_{\{3\}}$, $B_{\{1,3,4\}}$ and $B_{\{1\}}$. Finally, the set of all non-empty regions associated with the partition is given below with their sign-vectors:

$$B_{\varnothing}=(-1,-1,-1,-1),B_{\{2\}}=(-1,+1,-1,-1),B_{\{4\}}=(-1,-1,-1,+1),$$

$$B_{\{1,2\}}=(+1,+1,-1,-1),B_{\{2,4\}}=(-1,+1,-1,+1),B_{\{3,4\}}=(-1,-1,+1,+1),$$

$$B_{\{1,2,3\}}=(+1,+1,+1,-1),B_{\{2,3,4\}}=(-1,+1,+1,+1) \text{ and } B_{\{1,2,3,4\}}=(+1,+1,+1,+1).$$

4. A FORMULA FOR THE NUMBER OF REGIONS

The goal of this section is to develope a formula for the number of nonempty regions in terms of the number of hyperplanes and the cardinalities of integer sets corresponding to kinds of degenaracy classes : parallelicity and concurrency classes. The formula obtained here is useful for investigations about computational complexity of numerical methods in PWA analysis of non-linear circuits and systems [10-11] and is of importance in combinatorial geometric point of view.

It was shown in section 2 that a region B_I is not empty if and only if there exists no basic solution to $y^T A=0$ satisfying the angle condition $y^T b\geq 0$ in $\mathbf{R}_I$. Thus, the linear partition problem of $\mathbf{R}^n$ was transformed into another space $\mathbf{R}^l$ where the orhants, containing at least one basic solution which satisfies the angle condition, correspond to the nonempty regions of $\mathbf{R}^n$. This transformation enables to count the regions by considering the basic solutions only. It is clear that the basic solutions of $y^T A=0$ are related to the linear dependancy of the normal vectors α_i of the hyperplanes. On the other hand, the orthogonality of a basic solution y with the vector b (which implies both of the basic solutions y and -y satisfy the angle condition) means that the hyperplanes, whose normal vectors are linearly dependent, intersect in a k-flat. These observations show that the following definitions of parallelicity and concurrency classes will be useful for counting nonempty regions.

Note that if all order of parallelicity and concurrency classes, associated with a linear partition, are empty then this partition will be called nondegenerate.

Definition 1. : Given a linear partition $\Pi := \{ H_1, H_2, \ldots, H_l \}$, $\Omega_k = \{ \Omega_{k,1}, \Omega_{k,2}, \ldots, \Omega_{k,\#\Omega_k} \}$ is a class of k-parallelicity for $k \in \{ 1,2,\ldots,n-1 \}$ if

i). $\emptyset \neq \Omega_{k,i} \subseteq \Pi$ for all i,

ii). for each $\Omega_{k,i} \in \Omega_k$ there exists a k-flat F_u such that for each $H_p \in \Omega_{k,i}$ either $F_u \subseteq H_p$ or $F_u \cap H_p = \emptyset$,

iii). for $i \neq j$ $\Omega_{k,i} \subsetneq \Omega_{k,j}$,

iv). for each $\Omega_{k,i} \in \Omega_k$; $\#\Omega_{k,i} > n-k$

v). if a set of hyperplanes in Π satisfies the conditions in i)–iv) , it is an element of Ω_k . □

For instance, a bundle of parallel hyperplanes forms a class of (n-1)-parallelicity and three nonparallel planes in $\mathbf{R}^3$ not intersecting at a point form a class of 1-parallelicity.

Note 3. : It should be observed that $\bigcap_{some\ i} \Omega_{k,i}$ may not be empty. □

Definition 2. : A class of k-parallelicity $\Phi_k = \{ \Phi_{k,1}, \ldots, \Phi_{k,\#\Phi_k} \}$ is a class of k-concurrency for $k \in \{ 0,1,\ldots,n-1 \}$ if the condition ii). in Definition 1 is satisfied as follows :

ii). For each $\Phi_{k,i} \in \Phi_k$ there exists a unique k-flat F_u such that $F_u \subseteq H_p$ for all $H_p \in \Phi_{k,i}$. □

For instance, 4 lines passing through a point form a 0-concurrency in the plane and three planes intersecting in a line form a 1-concurrency in the 3-dimensional space.

Note 4. : For $k \geq 1$, some i and j , $\Phi_{k+1,i} \cap \Phi_{k,j}$ may not be empty. □

The relationships, between the basic solutions with the linear dependency of the normal vectors and between the angle condition with the consistency of the offset constans to the linear dependency among the normal vectors, have been mentioned above. Via the Definitions 1 and 2 one can construct the similar relationships among the basic solutions with the parallelicity classes and among the angle condition with the concurrency classes. By virtue of Definition 3 , a bridge among the basic solutions and the integer subsets of L will be established. Thus, the linear partition problem , which has been transformed to the subdivision of $\mathbf{R}^l$ into the orhants $\mathbf{R}_I$, will be transferred to the partition of the integer set L into several subsets. The number of regions will be obtained by counting the elements of these integer subsets of L which are derived from the concurrency and the parallelicity classes as in the Definition 4.

Definition 3. : A solution $y(\Lambda)$ of $y^T A = 0$ is called to be dependent on $\Lambda \subseteq L$ if there is a $\Lambda \subseteq L$ with a unique mutually exclusive decomposition $\Lambda = \Lambda^+ \cup \Lambda^-$,such that

$y_i(\Lambda) = 0$ for *all* $i \in L - \Lambda$, $y_i(\Lambda) > 0$ for *all* $i \in \Lambda^+$, $y_i(\Lambda) < 0$ for *all* $i \in \Lambda^-$ □ (13)

In Theorem 4, the number regions will be derived by using three integer sets : Γ , $\Gamma(\Omega_{k,i})$ and $\Gamma(\Phi_{k,i})$ in (14).

Definition 4. : Define the integer sets Γ , $\Gamma(\Omega_{k,i})$ and $\Gamma(\Phi_{k,i})$ as follows :

$$\Gamma := \{\, I \subseteq L \mid \textit{there exists a } \Lambda \textit{ such that } \Lambda^{+} \subseteq I \,,\, \Lambda^{-} \cap I = \varnothing \,\} \tag{14.a}$$

here for each $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{n+1}\} \subseteq L$ there corresponds a solution $y(\Lambda)$.

$$\Gamma(\Omega_{k,i}) := \{\, I \subseteq L \mid I \in \Gamma \,,\ \textit{there exists a } \Lambda(\Omega_{k,i}) \textit{ such that } \Lambda^{+}(\Omega_{k,i}) \subseteq I \ ,\ \Lambda^{-}(\Omega_{k,i}) \cap I = \varnothing \,\} \tag{14.b}$$

here $\Lambda(\Omega_{k,i}) := \{\lambda_1, \lambda_2, \ldots, \lambda_{n-k+1}\} \subseteq \{\, s \mid H_s \in \Omega_{k,i} \,\}$ associated with $y(\Lambda(\Omega_{k,i}))$.

$$\Gamma(\Phi_{k,i}) := \{\, I \subseteq L \mid I \in \Gamma \,,\, I \in \Gamma(\Omega) \,,\ \textit{there exists a } \Lambda(\Phi_{k,i}) \textit{ such that either } \Lambda^{+}(\Phi_{k,i}) \subseteq I \,,\, \Lambda^{-}(\Phi_{k,i}) \cap I = \varnothing \ \textit{ or } \ \Lambda^{-}(\Phi_{k,i}) \subseteq I \,,\, \Lambda^{+}(\Phi_{k,i}) \cap I = \varnothing \,\} \tag{14.c}$$

here $\Gamma(\Omega) = \bigcup_{k,i} \Gamma(\Omega_{k,i})$ and $\Lambda(\Phi_{k,i}) := \{\lambda_1, \lambda_2, \ldots, \lambda_{n-k+1}\} \subseteq \{\, s \mid H_s \in \Phi_{k,i} \,\}$ corresponds to $y(\Lambda(\Phi_{k,i}))$. □

Theorem 4 gives a formula for the number of regions in terms of the cardinalities of the sets in (14) without any restriction on the existence of degeneracies. $l \geq n$ is assumed with no loss of generality, if not some simple modification can be made.

Theorem 4. [11] **:** The number of nonempty regions in a linear partition of $\mathbf{R}^n$ by l hyperplanes is given by

$$Num(l,n) = 2^l - \#\Gamma - \# \bigcup_k \bigcup_i \Gamma(\Omega_{k,i}) - \# \bigcup_k \bigcup_i \Gamma(\Phi_{k,i}) \quad \square \tag{15}$$

Note that there are some formulas also in the literature [2-5,7,15]. However, all of them are restricted to some special cases ; three dimensional space as in [4] , nondegenerate partitions as in [2] and some of them concern the upper bounds only [6,15]. The most general result avaliable in the literature [2] , which is valid nondegenerate cases only,
is

$$Num(l,n) = \sum_{i=0}^{n} \binom{l}{i} \tag{16}$$

The formula (16) can be obtained from (15) , as a special case, by taking all parallelicity and concurrency classes as empty and then by dropping the last two terms in (15). Observe that in nondegenerate cases the number of regions depends on the number of hyperplanes l and the dimension of space n only. As can be seen from (15), in degenerate cases the cardinalities of parallelicity and concurrency classes should be considered.

The formula (15) derived in Theorem 4 can not directly be used for the given degeneracy classes. However, a formula can also be given in terms of $\#\Omega_{k,i}$ and $\#\Phi_{k,i}$. This is equivalent to the problem of finding the subsets of the integer set L which they include certain subsets corresponding to the degeneracy classes. For the cases in which only first kind of degeneracy (parallelicity) exists, the following formula is conjectured here :

$$Num(l,n) = \sum_{i=0}^{n} \binom{l}{i} - \sum_k \sum_i \binom{\#\Omega_{k,i}}{n-k+1} + \sum_k \sum_{i,j,\, i \neq j} \binom{\#\Omega_{k,i} \cap \Omega_{k,j}}{n-k+1}$$

$$-\sum_{k} \quad \sum_{i\,j\,p\,,i\neq j\neq p\neq i} \begin{bmatrix} \#\Omega_{k,i}\cap\Omega_{k,j}\cap\Omega_{k,p} \\ n-k+1 \end{bmatrix} + \ldots\ldots\ldots\ldots\ldots \tag{17}$$

(17) includes (16) since in the nondegenerate case $\Omega_{k,i}=\varnothing$ for all k, i.

The following example will be useful for understanding the above results.

Example 2. : Consider the linear partition $\Pi=\{H_1,H_2,H_3,H_4\}$ in Example 1 again. Using the definition of parallelicity and concurrency classes , the sets Γ , $\Gamma(\Omega)$ and $\Gamma(\Phi)$ needed to compute the number of regions are obtained below. The family of 1-parallelicity is $\Omega_1=\{\ \Omega_{1,1}\ \}=\{\ H_1,H_2\ \}$ and the family of 0-concurrency is $\Phi_0=\{\ \Phi_{0,1}\ \}=\{\ H_1,H_3,H_4\ \}$. Basic solutions of $y^T A=0$ are $y(\Lambda_1)=y(\{1,2\})=[\ 1\ -1\ 0\ 0\ ]$, $y(\Lambda_2)=y(\{1,3,4\})=[\ 1\ 0\ -1\ 2\ ]$, $y(\Lambda_3)=y(\{2,3,4\})=[\ 0\ -1\ 1\ -2\ ]$ and their opposites in sign. But only $y(\Lambda_1)$, $y(\Lambda_2)$, $-y(\Lambda_2)$, and $y(\Lambda_3)$ satisfy the angle condition.

It should be pointed out that all solutions of $y^T A=0$ can be written as nonnegative combinations of these basic solutions, namely; they are placed only in the orhants containing at least one basic solution. Then, all Λ_i 's , which are partitioned according to the signs of solutions, are obtained by using Λ_1 , Λ_2 and Λ_3 :

$$\Lambda_1=\Lambda_1^+\cup\Lambda_1^-=\{1\}\cup\{2\}\ ,\ \ \Lambda_2=\Lambda_2^+\cup\Lambda_2^-=\{1,4\}\cup\{3\}\ ,\ \ \Lambda_3=\Lambda_3^+\cup\Lambda_3^-=\{3\}\cup\{2,4\}\ ,$$
$$\Lambda_4=\Lambda_4^+\cup\Lambda_4^-=\{2,3\}\ ;\ \ \Lambda_5=\Lambda_5^+\cup\Lambda_5^-=\{1,3\}\cup\{2\}\ ,\ \ \Lambda_6=\Lambda_6^+\cup\Lambda_6^-=\{1\}\cup\{2,4\}$$
$$\text{and}\quad \Lambda_7=\Lambda_7^+\cup\Lambda_7^-=\{1,4\}\cup\{2\}\ .$$

The integer sets Γ , $\Gamma(\Omega)$ and $\Gamma(\Phi)$ are also obtained as follows ; for Λ_2 , Λ_3 , Λ_4 , Λ_5 and $\Lambda_6\in\{\ \lambda_1,\lambda_2,\lambda_3\ \}\subseteq L$

$$\Gamma=\{\ I\ |\ \textit{there exist a}\ \Lambda\ \textit{such that}\ \Lambda^+\subseteq I\ ,\Lambda^-\cap I=\varnothing\ \}=\{\ \{1,4\},\{1,4,2\},\{3\},\{3,1\},\{1\},\{1,3,4\}\ \}$$

; for $\Lambda_1=\Lambda(\Omega_1)=\{\ 1,2\ \}=\{\ s\ |\ H_s\in\Omega_1\ \}$,

$$\Gamma(\Omega_1)=\{\ I\ |\ I\in\Gamma\ ,\ \textit{there exists a}\ \Lambda(\Omega_1)\ \textit{such that}\ \Lambda^+(\Omega_1)\subseteq I\ ,\Lambda^-(\Omega_1)\cap I=\varnothing\ \}=\{\varnothing\}$$

; for $\Lambda_2=\Lambda(\Phi_0)=\{\ 1,3,4\ \}=\{\ s\ |\ H_s\in\Phi_0\ \}$,

$$\Gamma(\Phi_0)=\{\ I\ |\ I\in\Gamma\ ,I\in\Gamma(\Omega_1)\ ,\ \textit{there exists a}\ \Lambda(\Phi_0)\ \textit{such that}$$
$$\textit{either}\ \ \Lambda^+(\Phi_0)\subseteq I\ ,\Lambda^-(\Phi_0)\cap I=\varnothing\ ,\ \textit{or}\ \ ,\Lambda^-(\Phi_0)\subseteq I\ ,\Lambda^+(\Phi_0)\cap I=\varnothing\ \}=\{\ \{2,3\}\ \}$$

Thus, $Num(l,n)=Num(4,2)=2^l-\#\Gamma-\#\Gamma(\Omega)-\#\Gamma(\Phi)=2^4-6-0-1=9$ are found. □

5. CONCLUSION

An algebraic approach to determine and to count the nonempty regions of a linear partition has been given. Based on the method introduced, an algorithm capable of finding all regions and a formula for the number of regions have been developed. The method, the algorithm and the formula are valid in an arbitrary linear partition of $\mathbf{R}^n$ including all kinds of degenerate cases.

The results obtained are useful for PWA analysis of nonlinear circuits and systems since : i) determining all nonempty regions is a first step of some methods for finding all solutions of PWA resistive circuits [10,11], ii) computational complexity of combinatorial methods in PWA analysis [10,11] is proportional to the number of regions and iii) the number of equilibrium points of a PWA ordinary differential equations is closely related to the number of regions.

The method developed has potential applications to the computational geometry area. Since the formula, giving the number of regions, is valid for all kinds of linear partitions, it is a contribution to combinatorial geometry as well as to PWA analysis.

The following problems are subject to the future investigations; i) the comparation of the given algoritm with the ones available in the literature with respect to the storage and the computational efforts in a computer and ii) the derivation of a formula, for the number of regions, which can be calculated directly from the cardinalities of the parallelicity and concurrency classes.

REFERENCES

[1] Bieri ,H. and Nef ,W. , A recursive sweep-plane algorithm, determining all cells of a finite division of $\mathbf{R}^d$, Computing 28 , pp.189-198, 1982.

[2] Buck ,R.C. , Partition of space, Amer.math.monthly, Vol.50, pp.541-544, 1943.

[3] Zavlasky ,T. , Facing up to arrangements : face count formulas for partitions of space by hyperplanes, Memoirs amer.math.soc. 154, Providence, 1975.

[4] Alexanderson ,G.L. and Wetzel ,J.E. , Simple partitions of space, Mathematics magazine, Vol.51, pp.220-225, 1978.

[5] Polya ,G. , Induction and analogy in mathematics, Vol.1 of mathematics and plausible reasoning, Princeton university Press, 1954.

[6] Edelsbrunner ,H. , O'Rourke , J. and Seidel ,R. , Constructing arrangements of lines and hyperplanes with applications, in Proc.IEEE Annu.Symp.Foun.Comput.Sci. pp.83-91, November 1983.

[7] Winder ,R.O. , Partitions of N-space by hyperplanes, SIAM J. Appl.Math., 14, pp.814-818, 1966.

[8] Lee ,D.T. and Preparata ,F.P. , Computational geometry" A survey, IEEE Trans.on computers. Vol.C-33, No.12, December 1984.

[9] Rheinboldt ,W.C and Vandergraf ,J.S. , On piecewise-affine mapping in $\mathbf{R}^n$, SIAM J.Appl.Math.Vol. 29, No.4, December 1975.

[10] Güzeliş ,C. and Göknar ,İ.C. , Finding multiple solutions of piecewise-affine resistive circuits, in Proc.IEEE Int.Symp.on circuits and syst. pp.1229-1232, June 1988.

[11] Güzeliş ,C. , Piecewise-affine maps:structural characteristics and nonlinear circuit applications, Ph.D.thesis, Technical University of Istanbul, Turkey, November 1988.

[12] Nikaido ,H. , Convex structure and economic theory, Academic press, New York, 1968.

[13] Gale ,O. , The theory of linear economic models, Mc Graw-Hill, New York,1960.

[14] Strang ,G. , Linear algebra and it's applications, Academic press, New York, 1976.

[15] Grunbaum ,B. , Convex polytopes, Wiley, New York, 1967.

The Protocol Conversion Problem — Finding a Quotient of Specifications[1]

Kenneth L. Calvert
Simon S. Lam
Department of Computer Sciences
University of Texas at Austin

Abstract

The problem of deriving a *protocol converter* from given protocol specifications can be viewed as the problem of finding the "quotient" of two specifications. We discuss a theory of specifications of a class of finite-state systems, and describe an algorithm for solving "quotient" problems for the class. The algorithm is applied to an example conversion problem.

Introduction

Today it is possible to establish an information path between practically any two computer systems. Unfortunately, the ability to move bits between machines does not imply the capability for useful interaction; the connected systems must also have some common ground in the *protocols* they use. This situation is depicted schematically in Figure 1. System P_0 is designed to communicate with system P_1 according to protocol P, and together they provide some service to their users. A similar service is provided by Q_0 and Q_1, which are designed to use protocol Q. When P_0 needs to interact with Q_1, a *protocol mismatch* exists.

The obvious solution to the protocol mismatch problem is to modify one or both of the installations so that they use the same protocol. Unfortunately, this approach is not always practical. The term *protocol conversion* refers to techniques for overcoming a protocol mismatch *without* extensive modification of the existing systems. The approach we consider here involves placing an intermediary, or *protocol converter*, between P_0 and Q_1 (Figure 2); the converter "translates" between the two protocols, allowing P_0 and Q_1 to achieve some degree of useful interaction. It is not immediately clear, however, what constitutes a correct "translation" of one protocol into another [6], or when such a translation is possible.

Within the framework of an abstract model of protocols and services, protocol components and services can be precisely specified in an abstract form, and we can reason rigorously about whether a specified protocol provides a specified service. Within such a framework, the problem can be stated as follows: given specifications for protocols P and Q, and a specification of the service (S) to be provided by the conversion system, produce a specification for a converter, C, such that P_0, C, and Q_1 together implement the service defined by S, if such a converter exists. While this is a somewhat simplified view of the problem (in general, protocols may comprise several components, including services implemented by other protocols at lower levels in the architecture), it can be generalized; we later present an example involving multiple-component protocols.

Some methods of constructing a converter have been proposed in the literature [2, 7, 10]. These methods are "bottom-up," in that the existing protocols are used as the starting point for constructing the converter; they define *sufficient* conditions for a converter to be efficiently constructed. If one of these methods fails to yield a correct converter, one cannot, in general, conclude that none exists for the given inputs. In this paper, we present a "top-down" method of deriving a converter, in which the specification of the desired service is used as a starting point. In fact, our method can be used to solve the following general problem for a class of concurrent systems: given specifications for a service A and a component B, produce a specification for a component C, such that B and C together (i.e., interacting with each other) implement the service defined by A — if and only if such a C exists. In a sense, we want to "divide" A by B, and thus we call this a *quotient* problem.

Quotient problems arise in many areas of computing. Merlin and Bochmann [8] considered the "submodule construction" problem, and presented a solution based on a trace semantics. However, their method cannot deal with *progress* properties such as absence of deadlock. More recently, a semi-algorithmic method for CCS specifications has been described [11], and the "supervisor synthesis problem" for controlled discrete-event systems [12] has been discussed in the control-theory literature.

In what follows, we introduce an abstract model of protocol systems, describe the algorithm briefly, and give an example of how a protocol converter can be derived using the algorithm.

---

[1] work supported by the National Science Foundation under grant number NCR-8613338

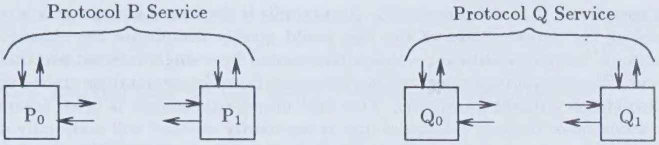


Figure 1: Protocols P and Q

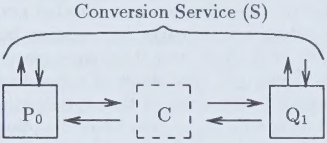


Figure 2: Protocol Converter

Specifications

We model a concurrent system, such as a protocol, as a set of interacting *components.* Components interact via named, synchronized *actions*; this form of interaction is similar to that of CSP [5] and other formalisms [9, 12]. Each component is specified in the form of a finite-state machine (FSM). We are concerned with two main ideas: a notion of *composition* of systems, i.e., viewing a system of interacting components together as a composite whole; and a notion of what it means for one system to *satisfy*, or *implement*, another. These concepts appear in the theory as a composition operator and a satisfaction relation on specifications. Together, they allow us to reason about whether a specified system of interacting protocol components correctly implements a specified service.

Definition. A *specification* is a tuple $(S, \Sigma, \mathcal{T}, \mathcal{I}, s_0)$, where S is a nonempty finite set of *states*; Σ is a finite set of *actions*; $\mathcal{T} \subseteq S \times \Sigma \times S$ is the *external transition relation*; $\mathcal{I} \subseteq S \times S$ is the *internal transition relation*; and $s_0 \in S$ is the distinguished *initial state.*

Thus, a specification is a finite-state machine. The set Σ of actions completely defines a component's interface with the environment (by "environment," we mean its users or other systems with which it is composed). The actions of the interface are the *only* way components can interact; intuitively, they model an exchange of information or handshake across the interface, possibly involving a state change on both sides. In a protocol or service specification, actions are abstractions of relatively complex occurrences such as submission of a message for transmission, or expiration of a timer.

The relations $\mathcal{T}$ and $\mathcal{I}$ define the possible *state changes* of the system. Each state change in $\mathcal{T}$ has an associated action in Σ; these define how the system's state is affected by interaction with its environment. If (s, e, s') is in $\mathcal{T}$, we say e is *enabled* in s, and write $s \xrightarrow{e} s'$. We refer to the transition and its associated action as an "event." Whenever the system is in state s, and the action e is also enabled in the environment, e may occur, and the state of the system simultaneously changes to s'. It is important to realize that external events are under the exclusive control of *neither* side of the interface, but can only occur when enabled on *both* sides. If multiple events are enabled in a state s, then the system cannot prevent any of them from occurring whenever it is in state s.

The relation $\mathcal{I}$ defines *internal* state changes that may occur unobserved, *without* environmental interaction. When (s, s') is in $\mathcal{I}$, we write $s \rightarrow s'$. Internal transitions introduce *nondeterminism* into specifications, by allowing some state transitions to occur under the exclusive control of one side of the interface. (N.B. "nondeterminism" here refers to the possibility of the same sequence of observable actions leading to different states, and *not* to the possibility of different actions being enabled at the same state.) Nondeterminism complicates the theory, but plays at least two important roles in specifications. In specifying a service, it can represent a choice among multiple acceptable behaviors. For example, if the environment provides an erroneous input, the acceptable behaviors might include ignoring it, or responding with an error message. This kind of nondeterminism allows high-level descriptions of desired behavior to avoid unnecessary overspecification. The choice among the acceptable behaviors only needs to be made once, when designing an implementation.

Nondeterminism is also useful as an abstraction mechanism in describing an implementation. Here, internal transitions model low-level behavior that may be random *or* deterministic, but which would add too much complexity if modeled explicitly. An example is the loss of a message in a communication channel: modeling the actual causes of the loss would greatly complicate the channel specification. Instead, the chain of events constituting a loss is represented by a single internal transition, which may or may not occur. The composition operator, in abstracting from the interactions among the components of a system, introduces internal transitions. This kind of nondeterminism is often assumed to be *fair*, meaning that an event or internal transition that is repeatedly enabled will eventually occur. Instead of having explicit fairness requirements attached to each transition, we make certain assumptions about fairness; the result is a considerable simplification of the theory at a relatively small cost in generality. These assumptions are discussed below, when we introduce the satisfaction relation.

It is sometimes helpful to think of a specification as a directed graph with labeled edges. The label of an edge, if any, is its corresponding action; edges representing transitions in $\mathcal{I}$ have no label. In what follows, we use upper case letters A, B, C, and D to name specifications. Components of different specifications are distinguished by subscripts. The states of a specification are represented by (primed) lower-case italic letters corresponding to the name of that specification: a and a' are states of A, etc. When the context includes multiple specifications, the letter denoting a state makes it clear to which specification it belongs, so that when we write $a \xrightarrow{e} a' \wedge b \xrightarrow{e} b'$, it should be clear that one transition is defined in $\mathcal{T}_{\mathrm{A}}$, while the other is in $\mathcal{T}_B$.

Composition of systems involves making each a part of the other's environment, and viewing them together as a composite whole; their interactions with each other are synchronized and hidden from the rest of the environment. This operation is modeled by a binary composition function || on specifications.

Definition. For any specifications A and B, (A||B) is given by:

$$
\begin{aligned}
S_{(\mathrm{A}\|\mathrm{B})} &= S_{\mathrm{A}} \times S_{\mathrm{B}} \\
\Sigma_{(\mathrm{A}\|\mathrm{B})} &= (\Sigma_{\mathrm{A}} \cup \Sigma_{\mathrm{B}}) - (\Sigma_{\mathrm{A}} \cap \Sigma_{\mathrm{B}}) \\
\mathcal{T}_{(\mathrm{A}\|\mathrm{B})} &= \{\, (\langle a,b\rangle, e, \langle a',b'\rangle) : e \in \Sigma_{(\mathrm{A}\|\mathrm{B})} \wedge ((a = a' \wedge b \xrightarrow{e} b') \vee (b = b' \wedge a \xrightarrow{e} a'))\,\} \\
\mathcal{I}_{(\mathrm{A}\|\mathrm{B})} &= \{\, (\langle a,b\rangle, \langle a',b'\rangle) : (b = b' \wedge a \rightharpoondown a') \vee (a = a' \wedge b \rightharpoondown b') \vee \\
&\qquad (\exists e : e \in \Sigma_{\mathrm{A}} \cap \Sigma_B \wedge a \xrightarrow{e} a' \wedge b \xrightarrow{e} b')\,\} \\
\langle a,b\rangle_0 &= \langle a_0, b_0\rangle
\end{aligned}
$$

We shall be concerned only with the portion of the state space of any FSM that is reachable from the initial state. Each internal transition of the composite comes from one of two sources: an internal transition in one of the components, or a synchronized event of the interface between them that becomes hidden in the composition. Note that interface events that are not enabled in both components do not appear in the composite.

Our definition of *satisfaction* of one specification by another has two parts, one dealing with *safety* ("what *can* happen"), and one dealing with *progress* ("what *will* happen"). Thus, "B satisfies A" if and only if B has all the safety and progress properties that A has. Both parts are defined using the notion of a *trace*.

A *trace* is a sequence of actions in Σ, and represents a possible behavior of the system as it might be observed by its environment. In terms of the directed graph structure, we say a trace *corresponds to* the sequence of labels along a finite directed path in the graph. We associate a particular set of traces with each specification, namely those corresponding to paths in the graph beginning at the initial state. This set includes all *possible* behaviors of the system, and thus captures all of its safety properties. Note that trace sets are prefix-closed, and hence the empty trace, denoted by ε, is a possible behavior of every system. (*Notation:* the letters t, r, q, t', etc. will be used to represent and quantify over arbitrary traces, while e denotes an arbitrary event. Events are treated as traces of length one, and concatenation is denoted by juxtaposition: te is a trace ending with event e. For specification A, we write A.t for the predicate "t is a trace of A.")

Definition. The relation $\overset{*}{\rightharpoondown}$ is the reflexive and transitive closure of $\mathcal{I}$. Thus, $s \overset{*}{\rightharpoondown} s'$ means s' is reachable from s via zero or more internal transitions.

Definition. Every specification defines a relation "$\rightarrow$," which is the least relation satisfying, for any states s, s', s'', trace t, and event e:

- $s \xrightarrow{\varepsilon} s$.

- $s \xrightarrow{t} s' \wedge s' \xrightarrow{*} s'' \Rightarrow s \xrightarrow{t} s''$.
- $s \xrightarrow{t} s' \wedge s' \xrightarrow{e} s'' \Rightarrow s \xrightarrow{te} s''$.

Intuitively, $s \xrightarrow{t} s'$ says that there is a path from s to s' corresponding to trace t.

Definition. For any specification A, A.t is defined as $(\exists a : a_0 \xrightarrow{t} a)$.

The trace set interpretation yields a simple definition of satisfaction with respect to safety: "B satisfies A with respect to safety" means for any trace t, B.t implies A.t. Thus, every possible behavior of B is also a possible behavior of A.

Because both sides of an interface must cooperate in order for something to happen at the interface, our notion of *progress* deals with what actions are enabled in the system after any particular trace. Given this information, the environment can ensure that there is always *some* event enabled on both sides of the interface, and thus prevent deadlock. Intuitively, our definition of "B satisfies A with respect to progress" says the following: for any trace t of B, (and A), and any set E of actions such that A is guaranteed to have some action in E enabled after trace t, B is also guaranteed to have some action in E enabled after t. Thus any environment that always has an enabled action in common with A (and thus will not deadlock with A) will always have an enabled action in common with B as well. This notion of progress is similar to the "refusals" of Hoare [5], and the "acceptance sets" of Hennessey [4].

If there is at most one path corresponding to any trace (i.e., if the specification is *deterministic*), then the state of the system after any trace is uniquely determined, and so is the set of enabled actions after any trace. If nondeterminism is present, things are more complicated. For one thing, a transition associated with an external event is *not* "guaranteed to be enabled" in any state with an outgoing internal transition, because it may be "pre-empted" by occurrence of the internal transition. So we can only talk about sets of events being enabled in states with no outgoing internal transitions. However, a trace may lead to a cycle of internal transitions, which could lead to continuous pre-emption of the events, if any, leaving the cycle. What can we say about which events the system is "guaranteed" to accept in this case?

Consider the following *fairness condition:* no external event or internal transition can be pre-empted infinitely many consecutive times. Under this condition, if there are events that "break" a cycle of internal transitions, eventually one of them must occur. Thus, a cycle of internal transitions can be regarded as a single state for the purposes of defining the set of enabled events; the set of events enabled in a state on such a cycle is the set of all events enabled in any state on the cycle.

We call the set of states on a cycle of internal transitions a "sink set" if there are no internal transitions leaving the set except those of the cycle. The importance of sink sets is this: in a component satisfying the fairness condition, once a state in a sink set is reached, some external transition leaving the sink set (if there is one) will eventually occur, because they cannot all be ignored forever. We write *sink*.s to indicate that state s is a member of a sink set. It is clear that $sink.s \equiv (\forall s' : s \xrightarrow{*} s' \Rightarrow s' \xrightarrow{*} s)$.

Definition. $\tau.s$ denotes the set of external events associated with transitions enabled in state s: $e \in \tau.s \equiv (\exists s' : s \xrightarrow{e} s')$.

Definition. $\tau^*.s$ denotes the set of all external events enabled in any state internally reachable from s: $e \in \tau^*.s \equiv (\exists s' : s \xrightarrow{*} s' \wedge e \in \tau.s')$. Intuition: $\tau^*.s$ contains all states that *may* occur next if the system's current state is s.

For a state with no outgoing internal transitions, the set of actions "guaranteed to be enabled" when the system is in that state is $\tau.s$. For a state in a sink set, the set of events "guaranteed to be enabled" when the system is at that state is $\tau^*.s$, provided the fairness condition is satisfied.

In defining "B satisfies A" for specifications B and A, we view B as the specification of an implementation, and assume that it satisfies the fairness condition. On the other hand, we regard A as a service specification, and assume that all A has *no* fairness requirements associated with its internal transitions. We further assume that A is given in a certain *normal form*. A specification is defined to be in normal form exactly if

(i) no state has both internal and external transitions leaving it.

(ii) For any s and s', $s \xrightarrow{*} s' \wedge s' \xrightarrow{*} s \Rightarrow s = s'$.

(iii) For any states s, s', s'', $\hat{s}$, and $\hat{s}'$: $s \xrightarrow{*} s' \wedge s \xrightarrow{*} s'' \wedge s' \xrightarrow{e} \hat{s} \wedge s'' \xrightarrow{e} \hat{s}' \Rightarrow \hat{s} = \hat{s}'$.

If A is in normal form, then for each trace t such that A.t, there is a unique state a such that the transition associated with the last action in t terminates at a. This state is denoted by $\psi_{\mathrm{A}}.t$; $\psi_{\mathrm{A}}.\varepsilon$ is defined to be a_0. Thus, normal form allows nondeterminism, but "focuses" it so that after any trace, the state of the system is "localized."

Definition. For states a and b (of specifications A and B, respectively), the predicate *prog.a.b* is defined to mean:

$$(\exists a' : a \xrightarrow{*} a' \wedge sink.a' \wedge \tau^*.a' \subseteq \tau^*.b)$$

We now define "B satisfies A with respect to progress" as

$$(\forall t, b : (b_0 \xrightarrow{t} b \wedge sink.b) \Rightarrow prog.(\psi_A.t).b)$$

This definition implies that any environment that does not deadlock with A after t cannot deadlock with B after t. It turns out that because a sink set is reachable from every state, the above formula is equivalent to $(\forall t, b : b_0 \xrightarrow{t} b \Rightarrow prog.(\psi_A.t).b)$, and thus, we may ignore sink sets when actually using this definition.

Definition. For A in normal form, with no fairness requirements, and B satisfying the fairness condition, "B satisfies A" if and only if

(i) $\forall t : \mathrm{B}.t \Rightarrow \mathrm{A}.t$. (Safety)

(ii) $\forall t, b : b_0 \xrightarrow{t} b \;\Rightarrow\; prog.(\psi_A.t).b$ (Progress)

An Algorithm for Computing a Quotient of Specifications

Let *Int* and *Ext* be disjoint sets of actions. We are given specifications A and B such that $\Sigma_{\mathrm{A}} = Ext$ and $\Sigma_{\mathrm{B}} = Int \cup Ext$, and must produce C such that $\Sigma_{\mathrm{C}} = Int$, and B||C satisfies A, or show that no such specification exists. The events in *Int* constitute the interface between B and C. The events in *Ext* are B||C's interface to the environment as well as A's interface. In terms of Figure 2, B corresponds to $\mathrm{P}_0||\mathrm{Q}_1$, and A corresponds to the service specification. The event set *Ext* is the interface between the user and the service, and *Int* represents the interactions — messages that may be sent and received — between the peers of protocols P and Q.

In computing C, we deal with safety and progress in two sequential phases. In the first phase, we build up the state set and transition relation of C inductively, beginning with the initial state. The result is a specification with the largest trace set consistent with safety of B||C. In the second phase, we iteratively remove states from this specification that do not have sufficient outgoing external transitions to prevent a possible progress violation. When this phase terminates, if C has a nonempty state space, then it is a solution, and moreover it is a *maximal* solution in the sense that, for any other solution D, we have $(\forall r : \mathrm{D}.r \Rightarrow \mathrm{C}.r)$. The following presentation is brief; the interested reader is referred to [3] for further detail.

We first define the projection functions i and o. If t is a trace of B (a sequence of events in $Int \cup Ext$), then $i.t$ is the projection of t onto its interface with C (i.e, *Int*), and $o.t$ is the projection onto its interface with the environment (*Ext*). The two functions are defined as follows:

$$\begin{aligned} i.\varepsilon &= \varepsilon \\ i.(te) &= \begin{cases} i.t & \text{if } e \in Ext \\ (i.t)e & \text{if } e \in Int \end{cases} \end{aligned} \qquad\qquad \begin{aligned} o.\varepsilon &= \varepsilon \\ o.(te) &= \begin{cases} o.t & \text{if } e \in Int \\ (o.t)e & \text{if } e \in Ext \end{cases} \end{aligned}$$

Definition. We say a trace r over actions in *Int* is *safe*, and write *safe*.r, if every trace of B that "matches" r is a trace of A when projected on *Ext*: $safe.r \equiv (\forall t : (i.t = r \wedge \mathrm{B}.t) \Rightarrow \mathrm{A}.(o.t))$.

Note that *safe*.re does not imply *safe*.r, and that r is trivially safe if no trace of B matches r.) It can be shown that for any C, $(\forall q : (\mathrm{B}||\mathrm{C}).q \Rightarrow A.q)$ if and only if $(\forall r : \mathrm{C}.r \Rightarrow safe.r)$. (Recall that $(\mathrm{B}||\mathrm{C}).q$ means q is a trace of B||C). It follows that a quotient (with respect to safety) exists if and only if *safe*.ε, because ε is a trace of every specification.

In constructing the state space of C, we shall associate with each state information about the possible current state of B and the externally-observable portion of the trace that led to it. This information is encoded as a set of (a, b) pairs, and we define a one-to-one correspondence *tag* between states of C and such sets. Moreover, we define a mapping h from traces over Int to such sets of (a, b) pairs, and then ensure that, for each r and c, $c_0 \xrightarrow{r} c \Rightarrow (tag.c = h.r)$.

Definition. $(a, b) \in h.r \equiv (\exists t : i.t = r \wedge b_0 \xrightarrow{t} b \wedge a = \psi_A.(o.t))$

We can use the sets $h.r$ for each r to check safety of the possible traces of C inductively, with the help of a predicate, *ok*, which we now define.

Definition. For any set J of (a, b) pairs such that $J = h.r$ for some trace r, $ok.J$ is equivalent to $(\forall a, b : (a, b) \in J \Rightarrow (\tau.b \cap Ext) \subseteq \tau^*.a)$. Intuitively, $ok.J$ says that for every pair (a, b) in J, any event in Ext that is enabled in b is also enabled in a state reachable from a. Note that $ok.J$ is easily checked by examination of J and the specifications A and B.

The following properties are consequences of the foregoing definitions. For any trace r over Int, and any action $e \in Int$:

$$ok.(h.\varepsilon) \Rightarrow safe.\varepsilon \tag{1}$$

$$safe.r \wedge ok.(h.re) \Rightarrow safe.re \tag{2}$$

$$safe.r \Rightarrow ok.r \tag{3}$$

We begin construction of C by computing $h.\varepsilon$, and checking $ok.(h.\varepsilon)$. If $ok(h.\varepsilon)$ holds, we can create an initial state c_0 and set $tag.c_0 = h.\varepsilon$; if not, no quotient exists. Then, if we have a way of computing $h.re$ from $h.r$, we can iterate, adding states and transitions until closure is achieved. Termination of this process is guaranteed by the finiteness of the range of h: the number of distinct sets of (a, b) pairs is finite.

The function φ maps a set J of (a, b) pairs and an event e to another set of pairs, such that $h.r = J$ implies $\varphi.(J, e) = h.re$. It is defined to be the set of all pairs (a, b) such that $(\exists a', b', t : (a', b') \in J \wedge i.t = e \wedge b' \xrightarrow{t} b \wedge a' \overset{e}{\mapsto} a)$, where the relation $a' \overset{e}{\mapsto} a$ is similar to $\rightarrow$, but has the added characteristic that $a' = \psi_A.q$ implies $a = \psi_A.qe$ (recall that A is in normal form).

We define $\Sigma_C = Int$, and $\mathcal{I}_C = \emptyset$. Thus, C has no internal transitions. The initial state space and external transition relation are computed by the following algorithm:

```
$S_C := \emptyset$; $new := \emptyset$;
$tag.c_0 := h.\varepsilon$;
if $ok.(tag.c_0)$ then $new := \{c_0\}$;
while $new$ is not empty
    select $c$ in $new$;
    for each $e$ in $Int$:
        $J := \varphi(tag.c, e)$;
        if $ok.J$ then
            if $tag^{-1}.J \notin (S_C \cup new)$
                then create $c'$;
                    $tag.c' := J$;
                    add $c'$ to $new$;
                else $c' := tag^{-1}.J$
            add $c \xrightarrow{e} c'$ to $\mathcal{T}_C$;
    move $c$ from $new$ to $S_C$;
```

Upon termination of the safety phase, we have for any trace r over Int, action $e \in Int$, and states $a \in S_A$, $b \in S_B$, and $c \in S_C$:

$$C.\varepsilon \equiv ok.(h.\varepsilon) \tag{4}$$

$$ok.(tag.c) \tag{5}$$

$$c_0 \xrightarrow{r} c \Rightarrow tag.c = h.r \tag{6}$$

$$C.r \wedge ok.(h.re) \Rightarrow C.re \tag{7}$$

$$(a, b) \in tag.c \equiv (\exists q : q \in Ext^* \wedge \langle b_0, c_0 \rangle \xrightarrow{q} \langle b, c \rangle \wedge a = \psi_A.q) \tag{8}$$

Theorem 1. Upon termination of the above portion of the algorithm, (i) every trace of C is safe. (ii) If D is any specification such that every trace of D is safe, then every trace of D is a trace of C.

Proof Sketch. Both parts are proved by induction on an arbitrary trace r. For (i), the base case, $safe.\varepsilon$, follows from (1) and (4). For the induction step, consider a trace re. $\mathrm{C}.re$ implies $\mathrm{C}.r$ by prefix-closure of trace sets, and by the inductive hypothesis we have $safe.r$. Also, $\mathrm{C}.re$ means $c_0 \xrightarrow{re} c$ for some c, hence by (6) we have $tag.c = h.re$ for some c. This implies $ok.(h.re)$, by (5). From $safe.r$ and $ok.(h.re)$, we conclude $safe.re$ by (2). For the base case of (ii), $\mathrm{D}.\varepsilon$ implies $safe.\varepsilon$, which by (3) implies $ok.(h.\varepsilon)$, which implies $\mathrm{C}.\varepsilon$ by (4). For the inductive step, $\mathrm{D}.re$ implies $\mathrm{D}.r$, which implies $\mathrm{C}.r$ by inductive hypothesis. $\mathrm{D}.re$ also implies $safe.re$, which implies $ok.(h.re)$. By (7), $\mathrm{C}.r$ and $ok.(h.re)$ imply $\mathrm{C}.re$.

The progress phase of the algorithm starts with C as computed in the safety phase (if it has a nonempty state space) and identifies states of B||C in which it is possible that a progress violation of A can occur.

Definition. A state c of C is *bad* if and only if $(\exists a, b : (a, b) \in tag.c \wedge \neg prog.a.\langle b, c\rangle)$.

By the definition of satisfaction with respect to progress and (8) above, this definition implies that B||C satisfies A with respect to progress if and only if C contains no bad states. Because the definition of a bad state depends on $\tau^*.\langle b, c\rangle$, which depends on $\mathcal{T}_\mathrm{C}$, we must iteratively identify and remove bad states, and recalculate $\tau^*.\langle b, c\rangle$ for each b and c. The process terminates when there are no more bad states to remove. Note that removing the initial state is equivalent to removing all remaining states, because it makes them all unreachable.

```
if $\mathcal{S}_\mathrm{C} \neq \emptyset$ then
    repeat
        $save := \mathcal{S}_\mathrm{C}$;
        compute $\tau^*.\langle b, c\rangle$ for each $b, c$ pair;
        foreach $c \in \mathcal{S}_\mathrm{C}$:
            foreach $(a, b) \in tag.c$:
                if $\neg prog.a.\langle b, c\rangle$ then
                    mark $c$ bad;
        remove bad states and their
            associated transitions from $\mathcal{S}_\mathrm{C}$ and $\mathcal{T}_\mathrm{C}$;
    until $c_0$ is removed or $save = \mathcal{S}_\mathrm{C}$
```

Theorem 2. If D is such that B||D satisfies A, and c is marked bad at some point in the above phase of the algorithm, and $c_0 \xrightarrow{r} c$, then r is not a trace of D.

We omit the proof of this result, which depends upon the maximality part of Theorem 1, and implies that the second phase of the algorithm maintains that property. As a consequence, if the algorithm terminates with an empty state set, we can conclude that no quotient exists.

An Example

We now show the application of the algorithm to a simple example. The protocols in the examples are the venerable alternating-bit (AB) protocol [1], and a non-sequenced (NS) protocol, which attaches no sequence numbers to data messages. Each protocol provides delivery of data messages from a Sender entity to a Receiver, in spite of possible losses by the underlying transmission medium. The specifications of the AB Sender and Receiver are depicted in Figure 3. The "acc" and "del" events (abbreviating "accept" a message from the Sender and "deliver" a message to the Receiver, respectively) constitute the interface with the user, while other events are interactions with the channel. Events beginning with "−" represent passing a message interface into the channel, while "+" indicates removal of a message from the channel. The Sender (A_0) attaches a one-bit sequence number to each data message transmitted. Data messages are denoted by "d0" and "d1." The Receiver (A_1) uses this sequence bit to synchronize with the Sender and determine whether a received data message has already been delivered; each data message is delivered exactly once (for each time it is accepted for transmission). An acknowledgement message, denoted "a0" or "a1," is returned for each data message; it contains the sequence number of the last-delivered data message.

The NS protocol (Figure 4) has no sequence numbers; the Receiver (N_1) delivers every received data message. A data message is represented by "D." The Sender (N_0) repeatedly transmits the data until an acknowledgement "A" is received. While both protocols guarantee that a message will be delivered at least once, the NS protocol may deliver the same message multiple times, while AB delivers each exactly once. The service of NS is thus somewhat "weaker" than that of AB.

The configuration of the desired conversion system is shown in Figure 5, where the dotted box represents the converter that is to be computed if possible. The channels between Sender and Receiver are modeled as separate components of the system. They are assumed to be lossy, and both protocols use timeouts to detect and recover from lost messages. The channel specifications appear in Figure 6. Unlabeled transitions in these FSMs represent the possible loss of a message; after such a loss, a timeout event occurs at the Sender. Note that these timeouts never occur prematurely. Again, "–" indicates passage of a message into the channel, and "+" indicates a message leaving the channel, so these events match their counterparts in the protocol specifications. The specification of the desired service is shown in Figure 7. It is similar to the service provided by the AB protocol.

For the given inputs (i.e., the service specification of Figure 7 and the composite specification $A_0||ABchannel||NSchannel||N_1$), the quotient algorithm fails to produce a specification. It follows that no converter exists for the given components and desired service. The problem is that if a message is lost between C and N_1, a "deliver" event may or may not have occurred at the receiver. If C retransmits the message, two "deliver" might occur in a row. On the other hand, if no deliver has occurred, and C goes ahead and acknowledges the message, two "accepts" in a row may occur. If C does nothing, no further "del" or "acc" events will occur, which violates the progress part of the service specification, which says that one or the other must always be enabled. It is often this kind of conflict between safety and progress that precludes existence of a converter.

It is possible to "weaken" the service specification to allow multiple deliveries per accepted message, and thereby obtain a converter that provides the same service as NS. Another approach, illustrated here, is to require that the converter be co-located with the NS Receiver, as shown in Figure 8, and thus remove the possibility of loss between the converter and N_1. The input to the algorithm is as before, except that the NS channel is omitted from the composite. The converter produced by the algorithm for this problem is shown in Figure 9. Note that the "+D" and "–A" events match the same events in N_1, and denote passage of a data message from C to N_1, and an acknowledgement message from N_1 to C, respectively.

The dotted boxes in Figure 9 indicate a superfluous portion of the converter; the cycles passing through these states are harmless, but do nothing for overall system progress. This is a consequence of deriving the maximal converter. Removing such "useless" portions of the converter is computationally expensive and is best done by hand.

Conclusions

The work presented here shows that "quotient" problems can be solved algorithmically for a certain class of finite state specifications. The problem of finding a protocol converter is an example of such problems. By solving for the converter in a top-down fashion, we can detect when no converter exists for given protocols and desired service. This represents an extension of earlier work on deriving protocol converters. Our method also deals with progress properties such as absence of deadlock in a reasonable way.

The quotient problem is computationally hard; in fact, deciding whether a quotient exists for nondeterministic specifications in our model is PSPACE-complete, even if only safety is considered. The algorithm presented here has exponential time and space complexity in the worst case. However, the progress phase of the algorithm does not add significantly to its complexity (it is polynomial in the size of the quotient produced by the safety phase). A converter produced by this algorithm has a maximal set of traces, and thus may contain spurious sequences of events that do not affect correctness, but decrease efficiency.

Areas of interest for future work include characterization of classes of instances for which the "complexity" of the converter/quotient is small relative to that of the input; these are the cases where interposing a converter will be practical. Characterization of general problems for which no converter exists would also be useful.

References

[1] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson. A note on reliable full-duplex transmission over half-duplex lines. *Communications of the ACM*, 12(5), May 1969.

[2] Kenneth L. Calvert and Simon S. Lam. Formal methods for protocol conversion. *IEEE Journal on Selected Areas of Communications*, special issue on network interconnection, January 1990 (to appear).

[3] Kenneth L. Calvert and Simon S. Lam. Finding quotients of specifications. Technical report, University of Texas at Austin, Department of Computer Sciences, in preparation.

[4] Matthew Hennessey. *Algebraic Theory of Processes.* MIT Press, 1988.

[5] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1986.

[6] Simon S. Lam. Protocol conversion — correctness problems. In *Proceedings ACM SIGCOMM '86, Stowe, VT*, 1986.

[7] Simon S. Lam. Protocol conversion. *IEEE Transactions on Software Engineering*, 14(3):353–362, March 1988.

[8] Philip M. Merlin and Gregor v. Bochmann. On the construction of submodule specifications and communications protocols. *ACM Transactions on Programming Languages and Systems*, 5(1), Jan 1983.

[9] Robin Milner. *A Calculus of Communicating Systems.* Springer-Verlag, 1980.

[10] K. Okumura. A formal protocol conversion method. In *Proceedings ACM SIGCOMM '86, Stowe, VT*, 1986.

[11] Joachim Parrow. Submodule construction as equation solving in CCS. In *LNCS 287: Proceedings of Seventh Conference on Foundations of Software Technology and Theoretical Computer Science.* Springer-Verlag, 1987.

[12] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.

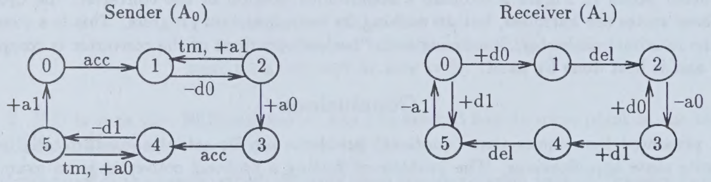

Sender (A₀)
0
1
2
3
4
5
acc
tm, +a1
-d0
+a0
+a1
-d1
acc
tm, +a0
Receiver (A₁)
0
1
2
3
4
5
+d0
del
-a1
+d1
+d0
-a0
del
+d1

Figure 3: Alternating Bit Protocol

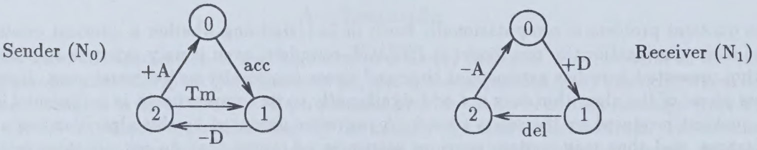

Sender (N₀)
0
1
2
+A
acc
Tm
-D
Receiver (N₁)
0
1
2
-A
+D
del

Figure 4: Non-sequenced Protocol

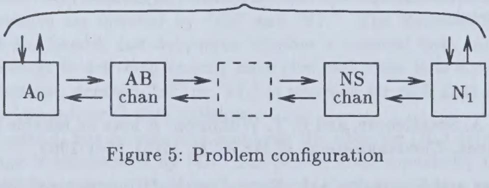

S
A₀
AB chan
NS chan
N₁

Figure 5: Problem configuration

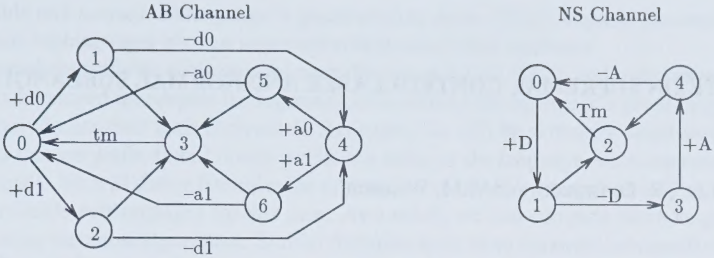

AB Channel
NS Channel
0
1
2
3
4
5
6
-d0
-a0
+d0
tm
+a0
+a1
-a1
+d1
-d1
0
1
2
3
4
-A
Tm
+D
+A
-D

Figure 6: Channel specifications

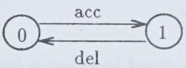


Figure 7: Desired Service (S)

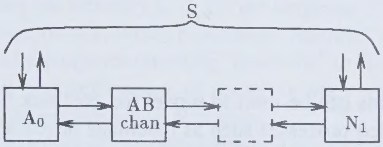

S
A_0
AB
chan
N_1

Figure 8: Alternative configuration

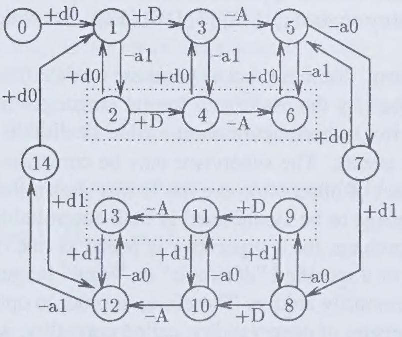

0
+d0
1
+D
3
-A
5
-a0
-a1
-a1
-a1
+d0
+d0
+d0
+d0
2
+D
4
-A
6
+d0
14
7
+d1
13
-A
11
+D
9
+d1
+d1
+d1
+d1
-a0
-a0
-a0
-a1
12
-A
10
+D
8

Figure 9: Converter for alternative configuration

A NOTE ON SUPREMAL CONTROLLABLE AND NORMAL SUBLANGUAGES

F. Lin[1], R. D. Brandt[1] and W.M. Wonham[2]

[1] Department of ECE, Wayne State University, Detroit, MI 48202
[2] Department of Electrical Engineering, University of Toronto,
Toronto, Ontario M5S 1A4, Canada

ABSTRACT

Supremal controllable and normal sublanguages have been shown to play an important role in supervisor synthesis. In this paper, we discuss computation of supremal controllable and normal sublanguages. We derive formulas for both supremal controllable sublanguages and supremal normal sublanguages when the languages involved are closed. As a result, those languages can be computed without applying recursive algorithms.

1. INTRODUCTION

Discrete event systems can be used to model the complex interactions of distributed, communicating, controlled processes such as machines or robots in a manufacturing system, programs or user's transactions in a computer system, and senders and receivers in a communication network. A system consisting of such interacting processes is usually described by traces (or sequences) of "events" that record significant qualitative changes in the "state" of the system.

The control of discrete event systems is quite different from the control of continuous variable systems and various approaches have been proposed. This paper is set in the framework of supervisory control ([1],[5],[7],[10],[12]). We will study the problem of synthesizing supervisors.

In supervisory control, discrete event systems are modeled by controlled automata, and their behaviors described by the associated formal languages (see appendix). Control is exercised by a supervisor, whose action is to enable or disable events in a distinguished subset of controllable events. The supervisor may be constrained to observe only events in a distinguished subset of observable events. It was shown that such a supervisor exists if and only if the language to be synthesized is both controllable and observable. Hence, the formal synthesis problem for a supervisor is posed as one of synthesizing the largest possible sublanguage of a specified "desirable" or "legal" language. This largest possible sublanguage is not necessarily unique. Therefore, in order to optimally solve the synthesis problem, a stronger version of observability, called normality, was introduced ([5],[7]). It has been shown in [7] that normality implies observability, and that a unique supremal controllable and *normal* sublanguage is guaranteed to exist. Hence, supremal controllable and normal sublanguages play an important role in supervisor synthesis.

Supremal controllable sublanguages were discussed in [14],[2],[3] and a recursive algorithm was developed to compute the supremal controllable sublanguage of a given language. But no formula has ever been derived. In this paper, we will be primarily concerned with languages that are prefix closed (every prefix of a string in the language is also contained in the language). We will derive formulas for supremal normal sublanguages as well as supremal controllable sublanguages for this case. As a result, we can compute those languages without using recursive algorithms. Similar formulas have been obtained independently by Kumar, Garg and Marcus ([4]) and are published in this same volume.

This paper is organized as follows. In Section 2, we discuss controllability, observability and normality of languages and the role of supremal controllable and normal sublanguages. In Section 3, we derive formulas for supremal controllable sublanguages and supremal normal sublanguages. The appendix reviews basic results in the theory of supervisory control.

2. CONTROLLABILITY AND OBSERVABILITY

Let Σ be a (finite) set of *events* and denote by Σ^* the set of all finite strings of events of Σ including the empty string ϵ. Let $\Sigma^+ = \Sigma^* - \{\epsilon\}$. A subset $L \subseteq \Sigma^*$ is a *language*. The *closure* of a language L is the set of all prefixes (initial segments) of strings in L. L is *closed* if $L = \overline{L}$. In this paper, we will assume languages are closed. We shall use the fact that if $B \subseteq \Sigma^*$ is closed, then for arbitrary $L \subseteq \Sigma^*$, the language $B - L\Sigma^*$ is closed.

Let M be a fixed language over Σ, which represents "feasible" or "physically possible" behaviors. Let $B \subseteq M$ be a language representing "desirable" or "legal" behavior. Methods for computing B from a set of specifications of operating rules was discussed in [9]. Our task here is to synthesize a supervisor when B is given.

We now introduce the concepts of controllability and observability ([1],[5],[7],[10]). It is natural to assume that some events are controllable, in the sense that their occurrence can be prevented; and some events are observable, in the sense that their occurrence can be observed. Thus, Σ is partitioned as

$$\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc} = \Sigma_o \dot{\cup} \Sigma_{uo}$$

where Σ_c is the set of *controllable events*, Σ_{uc} is the set of *uncontrollable events*, Σ_o is the set of *observable events*, and Σ_{uo} is the set of *unobservable events*.

An uncontrollable event ($\sigma \in \Sigma_{uc}$) is said to be *tolerable* if it is not physically possible for it to lead to an illegal behavior (i.e. $s\sigma \in L$ for all s such that $s\sigma \in M$). Intuitively, a language L is controllable if and only if all uncontrollable events are tolerable.

Definition 1 *A language $L \subseteq M$ is* controllable *(with respect to M) if*

$$(\forall s \in L)(\forall \sigma \in \Sigma_{uc}) s\sigma \in M \Rightarrow s\sigma \in L.$$

To define observability, let us first define $P : \Sigma^* \longrightarrow \Sigma_o^*$ to be

$$P\epsilon = \epsilon$$

$$P\sigma = \epsilon \, , \, \sigma \in \Sigma - \Sigma_o$$

$$P\sigma = \sigma \, , \, \sigma \in \Sigma_o$$

$$P(s\sigma) = (Ps)(P\sigma), \; s \in \Sigma^*, \; \sigma \in \Sigma$$

That is, P is a projection whose effect on a string $s \in \Sigma^*$ is just to erase the elements σ of s that do not belong to Σ_o.

In words, for an arbitrary pair of strings s and s', the predicate, $consistent(s, s')$, holds if and only if an arbitrary event σ should have the same enablement/disablement status after both s and s'. Then observability requires that if s and s' look the same ($Ps = Ps'$), then they must be consistent ($consistent(s, s')$). In other words, a language L is observable if unobservable events are insignificant in determining enablement/disablement status.

Definition 2 *A language* $L \subseteq M$ *is* observable *(with respect to* M*) if*

$$(\forall s, s' \in \Sigma^*) Ps = Ps' \Rightarrow consistent(s, s')$$

where

$$consistent(s, s') \Leftrightarrow (\forall \sigma) s\sigma \in L \land s' \in L \land s'\sigma \in M \;\Rightarrow\; s'\sigma \in L$$

If B is not controllable and observable, then the best we can do is to synthesize a largest controllable and observable sublanguage of B. However, this largest possible sublanguage is not necessarily unique ([7]). Therefore, in order to optimally solve the synthesis problem, a stronger version of observability, called normality, needs to be introduced.

A language L is normal if one can check whether a string $s \in M$ is legal ($s \in L$) by checking whether its projection Ps belongs to PL. In other words, information on occurrences of unobservable events is not needed in determining the legal status of a string.

Definition 3 *A language* $L \subseteq M$ *is normal (with respect to* M*) if*

$$(\forall s \in M) s \in L \Leftrightarrow Ps \in PL$$

Intuitively, one expects normality to be stronger than observability. This was proved in [7].

Theorem 1 *If* L *is normal then* L *is observable.*

We can also show that if $\Sigma_c \subseteq \Sigma_o$, i.e. every controllable event is also observable, then normality is equivalent to observability.

Theorem 2 *Under the condition* $\Sigma_c \subseteq \Sigma_o$*, if* L *is controllable and observable, then* L *is controllable and normal.*

Proof

Assume that L is nonempty, controllable and observable. We show that L is normal. Otherwise there exists a string t in M which does not belong to L but whose projection belongs to PL. Clearly, $t \neq \epsilon$. Let $s'\sigma$ be the shortest string among all such strings. Then

$$s' \in L \wedge s'\sigma \in M \wedge s'\sigma \notin L \wedge P(s'\sigma) \in PL$$

Now,

L is controllable

$$\Rightarrow \sigma \in \Sigma_c$$

$$\Rightarrow \sigma \in \Sigma_o, i.e., P(\sigma) = \sigma$$

This together with $P(s'\sigma) \in PL$ imply

$$(\exists s \in \Sigma^*) Ps = Ps' \wedge s\sigma \in L$$

Therefore,

$$s\sigma \in L \wedge s' \in L \wedge s'\sigma \in M \wedge s'\sigma \notin L$$

$$\Rightarrow \mathit{not\ consistent}(s, s')$$

Contradiction!

□

The class of normal sublanguages of M is algebraically better behaved than the observable sublanguages, as illustrated by the following theorem, which was proved in [7] and [14].

Theorem 3 *Let $B \subseteq M$ and consider the classes of languages*

$$C(B) = \{L \subseteq B : L \text{ is closed and controllable}\}$$

$$N(B) = \{L \subseteq B : L \text{ is closed and normal}\}$$

Then $C(B)$ and $N(B)$ are nonempty and closed under unions. Therefore, sup$C(B)$ exists in $C(B)$ and sup$N(B)$ exists in $N(B)$.

Write $CN(B)$ for $C(B) \cap N(B)$. Then it follows from Theorem 3 that $CN(B)$ is nonempty and closed under unions. Furthermore, sup$CN(B)$ exists in $CN(B)$. By Theorem 1, any language in $CN(B)$ is controllable and observable. Hence $CN(B)$ constitutes a natural and convenient class of sublanguages to be considered as candidates for the solution of our synthesis problem. In the next section, we will show how to compute sup$CN(B)$.

3. MAIN RESULTS

Let us first compute $\sup C(B)$. A recursive algorithm was developed in [14] to compute $\sup C(B)$ (for both closed and unclosed languages). But no formula has been given. When languages are closed we can derive a formula for $\sup C(B)$ as follows. Let $D_{uc} : \Sigma^* \to \Sigma^*$ be defined by

$$D_{uc}(s) = \text{the longest prefix of } s \text{ whose last event is controllable}$$

That is the effect of D_{uc} on a string s is to cut the "uncontrollable tail" of s. If s contains no controllable events, then $D_{uc}(s) = \epsilon$.

Intuitively, $M - B$ are illegal strings and $D_{uc}(M - B)$ are the strings that have the potential of becoming illegal (by adding uncontrollable events). Therefore, they have to be removed from B. Thus the formula for $\sup C(B)$ is given below, where Σ^* is concatenated to $D_{uc}(M - B)$ to make H closed.

Theorem 4

$$\sup C(B) = B - D_{uc}(M - B)\Sigma^*$$

Proof

Denote the right hand side by H. First, we show that $\sup C(B)$ contains H. Clearly H is closed and contained in B. Therefore, we only need to show that it is controllable. This is true because, otherwise, there exist s and σ such that

$$s \in H \wedge \sigma \in \Sigma_{uc} \wedge s\sigma \in M \wedge s\sigma \notin H$$

$$\Rightarrow s \in H \wedge \sigma \in \Sigma_{uc} \wedge s\sigma \in M \wedge (s\sigma \notin B \vee s\sigma \in D_{uc}(M - B)\Sigma^*)$$

$$\Rightarrow s \in H \wedge \sigma \in \Sigma_{uc} \wedge s\sigma \in M \wedge (s\sigma \in M - B \vee s\sigma \in D_{uc}(M - B)\Sigma^+)$$

Since either $s\sigma \in M - B$ or $s\sigma \in D_{uc}(M - B)\Sigma^+$ implies $s \in D_{uc}(M - B)\Sigma^*$, which contradicts $s \in H$.

Now we show that $\sup C(B)$ is a subset of H. Since $\sup C(B) \subseteq B$, we only need to show

$$D_{uc}(M - B)\Sigma^* \cap \sup C(B) = \emptyset$$

This is true because, otherwise, there exists s such that

$$s \in D_{uc}(M - B) \wedge s \in \sup C(B)$$

$$\Rightarrow (\exists u \in \Sigma_{uc}^*) su \in M - B \wedge s \in \sup C(B)$$

$$\Rightarrow (\exists u \in \Sigma_{uc}^*) su \notin B \wedge su \in M \wedge s \in \sup C(B)$$

$$\Rightarrow (\exists u \in \Sigma_{uc}^*)\, su \notin B \land su \in \sup C(B)$$

Contradiction!

□

Now let us compute $\sup N(B)$. Recall that the projection P erases all unobservable events. Intuitively, $M - B$ are illegal strings and $P^{-1}P(M - B)$ are the strings that look the same as illegal strings. Therefore, they have to be removed from B. Thus the formula for $\sup N(B)$ is given below, where Σ^* is concatenated to $P^{-1}P(M - B)$ to make H closed.

Theorem 5

$$\sup N(B) = B - P^{-1}P(M - B)\Sigma^*$$

Proof

Denote the right hand side by H. First, we show that $\sup N(B)$ contains H. Clearly H is closed and contained in B. Therefore, we only need to show that it is normal. This is true because, otherwise, there exists s such that

$$s \in M \land Ps \in PH \land s \notin H$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s \in M \land s' \in H \land s \notin H$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s \in M \land s' \in B \land s' \notin P^{-1}P(M - B)\Sigma^*$$
$$\land (s \notin B \lor s \in P^{-1}P(M - B)\Sigma^*)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s' \notin P^{-1}P(M - B)\Sigma^*$$
$$\land (s \in M - B \lor s \in P^{-1}P(M - B)\Sigma^*)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s' \notin P^{-1}P(M - B)\Sigma^* \land Ps \in P(M - B)\Sigma_o^*$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s' \notin P^{-1}P(M - B)\Sigma^* \land s' \in P^{-1}P(M - B)\Sigma^*$$

Contradiction!

Now we show that $\sup N(B)$ is a subset of H. Since $\sup N(B) \subseteq B$, we only need to show

$$P^{-1}P(M - B)\Sigma^* \cap \sup N(B) = \emptyset$$

This is true because, otherwise, there exists s such that

$$s \in P^{-1}P(M - B) \land s \in \sup N(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s' \in M - B \land s \in \sup N(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s' \notin B \land s' \in M \land s \in \sup N(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps = Ps' \land s' \notin B \land s' \in \sup N(B)$$

Contradiction!

□

Finally, we give a formula for $\sup CN(B)$. Using the formula, we can decompose the computation of $\sup CN(B)$ and apply the previous two formulas to compute $\sup CN(B)$.

Theorem 6

$$\sup CN(B) = M \cap P^{-1}\sup C_p(P\sup N(B))$$

where, for $D \subseteq PM$

$$C_p(D) = \{L \subseteq D : L \text{ is controllable w.r.t. } PM\}$$

Proof

Denote the right hand side by H. First, we show that $\sup CN(B)$ contains H. Clearly H is closed and normal. H is contained in B because

$$H \subseteq M \cap P^{-1}P\sup N(B) = \sup N(B) \subseteq B$$

It can also be shown that H is controllable (Proposition 3.1 of [6]). Therefore, $\sup CN(B) \supseteq H$.

Now we show that $\sup CN(B)$ is a subset of H. We only need to show

$$P\sup CN(B) \subseteq \sup C_p(P\sup N(B))$$

This is true because first,

$$\sup CN(B) \subseteq \sup N(B)$$

$$\Rightarrow P\sup CN(B) \subseteq P\sup N(B)$$

and secondly, $P\sup CN(B)$ is controllable w.r.t. PM. Because, otherwise, there exist s and σ such that

$$s \in P\sup CN(B) \wedge \sigma \in \Sigma_{uc} \cap \Sigma_o \wedge s\sigma \in PM \wedge s\sigma \notin P\sup CN(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps' = s \wedge s'\sigma \in M \wedge s \in P\sup CN(B) \wedge \sigma \in \Sigma_{uc} \cap \Sigma_o$$

$$\wedge s\sigma \notin P\sup CN(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps' = s \wedge s'\sigma \in M \wedge s' \in \sup CN(B) \wedge \sigma \in \Sigma_{uc} \cap \Sigma_o$$

$$\wedge s\sigma \in PM \wedge s\sigma \notin P\sup CN(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps' = s \wedge s'\sigma \in \sup CN(B) \wedge s\sigma \notin P\sup CN(B)$$

$$\Rightarrow (\exists s' \in \Sigma^*) Ps' = s \wedge P(s'\sigma) = s\sigma \in P\sup CN(B) \wedge s\sigma \notin P\sup CN(B)$$

Contradiction!

□

4. CONCLUSION

In this paper, we have shown that when the legal language is closed (contains all its prefixes), there are simple formulas for the supremal controllable sublanguage and the supremal normal sublanguage of the legal language. The closure condition is not restrictive in investigating "safety" properties of systems (e.g. a system should never leave a desired region), because in such cases, legal languages are in fact closed.

In investigating "liveness" properties of systems (e.g. a system should eventually accomplish some task from a prescribed set of completed or terminal tasks), the closure condition may not be satisfied. In such cases, the recursive algorithm developed in [14] is applicable and can be shown to be equivalent to a recursive application of the formula we have given for closed languages. In a sequel to this work, we will show how to recursively apply the formulas presented in this paper to find the supremal controllable and normal sublanguages for arbitrary legal languages.

ACKNOWLEDGEMENT

We like to thank Dr. Siu O'Young for his comments and contributions in implementing the new formulas with object-oriented computer software.

REFERENCES

[1] R. Cieslak, C. Desclaux, A. Fawaz and P. Varaiya, 1988. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control 33(3)*, pp. 249-260.

[2] H. Cho and S. I. Marcus. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. to appear in *Mathematical Systems Theory*.

[3] H. Cho and S. I. Marcus. On supremal languages of class of sublanguages that arise in supervisor synthesis problems with partial observations. *Math. Control Signal Systems* 2, pp. 47-69.

[4] R. Kumar, V. Garg and S. I. Marcus, 1989. Supervisory control of discrete event systems: supremal controllable and observable languages. *This Proceeding*.

[5] F. Lin, 1987. On Controllability and Observability of Discrete Event Systems. *Ph. D. Thesis*, Department of Electrical Engineering, University of Toronto.

[6] F. Lin and W. M. Wonham, 1988. Decentralized supervisory control of discrete-event systems. *Information Sciences, 44(3)*, pp. 199-224.

[7] F. Lin and W. M. Wonham, 1988. On observability of discrete event systems. *Information Sciences, 44(3)*, pp. 173-198.

[8] F. Lin and W. M. Wonham, 1988. Decentralized control and coordination of discrete event systems with partial observation. *Systems Control Group Report No. 8909*, Department of Electrical Engineering, University of Toronto.

[9] F. Lin, A. F. Vaz and W. M. Wonham, 1988. Supervisor specification and synthesis for discrete event systems. *International Journal of Control, 48(1)*, pp. 321-332.

[10] R. J. Ramadge and W. M. Wonham, 1987. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization, 25(1)*, pp. 206-230.

[11] P. J. Ramadge and W. M. Wonham, 1988. Modular supervisory control of discrete event systems. *Mathematics of Control, Signal and Systems, 1(1)*, pp. 13-30.

[12] P. J. Ramadge and W. M. Wonham, 1989. The control of Discrete Event Systems. *Proceedings of IEEE, 77(1)*, pp. 81-98.

[13] W. M. Wonham, 1988. A control theory for discrete-event systems. *Advanced Computing Concepts and Techniques in Control Engineering* (M. J. Denham and A. J. Laub Ed.), NATO ASI Series, F47, Springer-Verlag, Berlin, pp. 129-169.

[14] W. M. Wonham and P. J. Ramadge, 1987. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization, 25(3)*, pp. 635-659.

[15] H. Zhong, 1987. Control of discrete-event systems: decentralized and hierarchical control. *M. A. Sc. Thesis*, Department of Electrical Engineering, University of Toronto.

APPENDIX

In this appendix, we review and summarize the main results of supervisory control of discrete event systems. Abstractly, an uncontrolled discrete event system is taken to be an automaton, called *generator*

$$G = (\Sigma, Q, \delta, q_0, Q_m)$$

where $Q = \{q, q', ...\}$ is a set of states, and $\delta : \Sigma \times Q \to Q$, the transition function, is a partial function (pfn), defined at each $q \in Q$ for a subset of the $\sigma \in \Sigma$. Events are state transitions $q \xrightarrow{\sigma} q'$ where $\delta(\sigma, q) = q'$. The initial state is $q_0 \in Q$, and the subset of marker states is $Q_m \subseteq Q$. The latter serve to 'mark' the termination of 'tasks' or sequences of tasks of the underlying physical process that G is intended to model. $\delta : \Sigma \times Q \to Q$ is extended in the obvious way to $\delta : \Sigma^* \times Q \to Q$, where $\delta(\epsilon, q) := q$; and the language *generated* by G is

$$L(G) := \{s \in \Sigma^* : \delta(s, q) \text{ is defined}\}.$$

The language *marked* by G is

$$L_m(G) := \{s \in \Sigma^* : \delta(s, q_0) \in Q_m\}.$$

We make the natural assumptions that $L(G) \neq \emptyset$ and that $\overline{L}_m(G) = L(G)$.

Events are considered to occur spontaneously (no auxiliary forcing mechanism is postulated), asynchronously (i.e. without reference to a clock) and instantaneously.

Our objective is to design a controller (or *supervisor*) whose task is to enable and disable the controllable events based on a record of occurrences of the observable events such that the resulting closed loop system obeys some prespecified operating rules. Formally, a supervisor is a pair

$$S = (R, \psi)$$

where $R = (\Sigma, X, \xi, x_0, X_m)$ is an automaton called *recognizer* with $\xi : \Sigma \times X \to X$ (pfn) satisfying

$$(\forall \sigma \in \Sigma_{uo})\xi(\sigma, x) = x$$

and $\psi : \Sigma \times X \to \{0, 1\}$ is the *feedback map* satisfying

$$\psi(\sigma, x) = 1, \text{ if } \sigma \in \Sigma_{uc},\ x \in X$$

$$\psi(\sigma, x) \in \{0, 1\}, \text{ if } \sigma \in \Sigma_c,\ x \in X$$

R is considered to be driven externally by the stream of event symbols generated by G; while in turn, with R in state x, the transitions σ of G are subject to the control $\psi(\sigma, x)$. If $\psi(\sigma, x) = 0$ then σ is 'disabled' (prohibited from occurring), while if $\psi(\sigma, x) = 1$, then σ is 'enabled' (permitted but not forced to occur). In this way, there results a closed-loop feedback structure S/G, called the *supervised discrete-event system.* S/G is itself an automaton

$$S/G = (\Sigma, X \times Q, (\xi \times \delta)^{\psi}, (x_0, q_0), X_m \times Q_m)$$

where $(\xi \times \delta)^{\psi} : \Sigma \times X \times Q \to X \times Q$ is defined as

$$(\xi \times \delta)^{\psi}(\sigma, x, q) = \begin{cases} (\xi(\sigma, x), \delta(\sigma, q)) & \text{if } \psi(\sigma, x) = 1 \\ \text{undefined} & \text{if } \psi(\sigma, x) = 0 \end{cases}$$

A supervisor $S = (R, \psi)$ for G is *complete* if the three conditions

1. $s \in L(S/G)$
2. $s\sigma \in L(G)$, i.e., $\delta(s\sigma, q_0)$ is defined
3. $\psi(\sigma, \xi(s, x_0)) = 1$, i.e., σ is enabled at $\xi(s, x_0)$

   together imply that

4. $s\sigma \in L(S/G)$, i.e., $\xi(s\sigma, x_0)$ is defined.

It will always be assumed (or guaranteed) that S is complete. The behavior of the supervised discrete-event system is described by the languages $L(S/G)$, and $L_m(S/G)$. The existence condition of a supervisor is given in [5],[7],[10].

Theorem 7 *Let* L *be a nonempty sublanguage of* $L(G)$. *There exists a supervisor* S *such that* $L(S/G) = L$ *if and only if* L *is both controllable and observable.*

Various important control problems are discussed in this framework, which include modular supervision [11], decentralized supervision [6], [8], and hierarchical supervision [8],[15].

SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS: SUPREMAL CONTROLLABLE AND OBSERVABLE LANGUAGES

RATNESH KUMAR, VIJAY GARG and STEVEN I. MARCUS
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712

ABSTRACT

This paper studies the supervisor synthesis problem of a Discrete Event Dynamical System by way of *synchronous composition* of the plant and the supervisor and discusses some of the resulting controllability issues. An algorithm, that is computationally more efficient than the previously existing ones, is presented for the construction of the *minimally restrictive* supervisor and shown to be optimal. Closed form representation of the supremal controllable and of the supremal observable sublanguages is also presented.

1. INTRODUCTION

Ramadge and Wonham in their work [3] on (Discrete Event Dynamical Systems) DEDS have modelled a DEDS, also called a *plant* or a *generator*, by a State Machine (SM) , the *event* set of which is finite and is further partitioned into the sets of *controllable* and *uncontrollable* events. The *language* generated by such a SM is used as a model to describe the behavior of the plant at the *logical* level. The control task is formulated as that of synthesis of a *controller*, also called a *supervisor*. The way a supervisor exercises *open-loop control* over the plant is by disabling some of the controllable events in order that the plant may achieve a certain prescribed behavior. Supervisors designed for *closed-loop control*, so that none of the uncontrollable events that can occur in the plant behavior are prevented from occurring in the closed loop system, are called *complete*.

Since under the closed-loop control of a complete supervisor, all uncontrollable events that can occur in the plant, must also occur in the closed loop system, there may not always exist a complete supervisor for a given plant such that the closed loop system has a prespecified desired behavior. The question then arises as to how to construct a complete supervisor which is *minimally restrictive*, so that the closed loop system can engage in some maximal behavior and still maintain the prescribed behavioral constraint. Another problem related to the design of such supervisors is to reduce the *computational complexity* of finding the minimally restrictive supervisor [3],[4],[6].

In [3],[4] the supervisor is modelled as another SM together with a *feedback map* that is used to disable certain controllable events so that those events can no longer occur in the controlled plant behavior. We use the notion of the *synchronous composition* [2], [5] of the two machines, the plant and the supervisor, to describe the control achieved by the supervisor over the plant behavior. In this way we avoid the notion of having the feedback map. This simplifies the analysis of supervisory synthesis problems, as demonstrated by our treatment of some of the supervisory synthesis problems discussed below.

The computational complexity of the algorithm presented in [4] for constructing the minimally restrictive supervisor is quadratic [6] in the product of the number of states involved in the FSM realizations of the plant and that of the supervisor. We present an algorithm for determining the minimally restrictive supervisor that is an order of magnitude faster than the algorithm described in [4] and show the optimality of this algorithm. Next we provide closed form expressions for representing the supremal controllable and the supremal observable sublanguages and discuss the computational aspects of these expressions. Due to the lack of space we have not included some of the proofs in this paper and they can be found in [10].

2. NOTATION AND TERMINOLOGY

The DEDS that is to be controlled, also called the plant, is represented as a deterministic SM. Letting P denote the plant, it is represented as a 5-tuple [1]:

$$\mathrm{P} \equiv (X, \Sigma, \alpha, x_0, X_m)$$

where X denotes the states of the plant (X is finite, if P is a FSM); Σ denotes the *finite* event set; $\alpha : \Sigma \times X \to X$ is the *partial state transition function*; $x_0 \in X$ denotes the *initial* state of the plant; and $X_m \subseteq X$ denotes the *marked* states of the plant.

The behavior of the plant is described in terms of the set of strings of events called the *language* that the plant can generate. Formally, the languages *generated* and *recognized* or *marked* by P are denoted by L(P) and $L_m(P)$ respectively and are defined as:

$$\mathrm{L(P)} = \{\ s \in \Sigma^\star \mid \alpha(s, x_0)\ !\};\ L_m(P) = \{\ s \in L(P) \mid \alpha(s, x_0) \in X_m\}.$$

where $\Sigma^\star$ denote the set of all the finite strings of events belonging to Σ and the notation "!" is used to denote "is defined". The state transition function α is extended to the domain $\Sigma^\star \times X$ in the natural way. The language L(P) is *prefix closed* by its definition. Further, if the plant is given to be a FSM, then L(P) and $L_m(P)$ both are *regular* languages.

A supervisor for the given plant is another DEDS modeled as a deterministic SM (for a more general definition see [3]). Letting S denote the supervisor, it is represented as a 5-tuple:

$$\mathrm{S} \equiv (Y, \Sigma, \beta, y_0, Y_m)$$

where Y denotes the states of the supervisor; Σ denotes the finite event set, same as that of the plant; $\beta : \Sigma \times Y \to Y$ denotes the partial state transition map; $y_0 \in Y$ denotes the initial state of the supervisor; and $Y_m \subseteq Y$ denotes the marked states of the supervisor.

The supervisor executes *synchronously* with the plant and thereby controls the plant behavior. Let the synchronous composition [2] of the plant and the supervisor be denoted by the SM, P□S, where "□" represents the synchronous composition operator. The synchronous composition of the plant and the supervisor means that only those transitions which are allowed in both the plant P and the supervisor S, are allowed in the coupled SM, P□S, when run synchronously. Thus, if some of the transitions are not allowed in the supervisor then they also cannot occur in the plant. This is the way the supervisor controls the plant. Formally, P□S is another deterministic SM, represented by the 5-tuple:

$$\mathrm{P}\Box\mathrm{S} \equiv (Z, \Sigma, \gamma, z_0, Z_m)$$

where $Z = X \times Y$ is the state set of the coupled SM, P□S; Σ denotes the finite event set as before; $\gamma : \Sigma \times Z \to Z$ denotes the partial state transition map for P□S. Let $\sigma \in \Sigma$ and $(x, y) \in X \times Y = Z$; then we define[1]

$$\gamma(\sigma, (x, y)) \overset{\text{def}}{=} \begin{cases} (\alpha(\sigma, x), \beta(\sigma, y)) & \text{if } \alpha(\sigma, x)! \text{ and } \beta(\sigma, y)! \\ \text{undefined} & \text{otherwise} \end{cases}$$

$z_0 = (x_0, y_0)$ denotes the initial state of the coupled SM, P□S; and $Z_m = X_m \times Y_m$ denotes the marked states of the coupled SM.

Remark 2.1 We note that the transition function $\gamma(\sigma, (x, y))$ is defined iff $\alpha(\sigma, x)$ and $\beta(\sigma, y)$ both are defined.

Lemma 2.2 Let L(P□S) be the language generated and $L_m(P\Box S)$ the language marked by the P□S; then L(P□S) = L(P) ∩ L(S), and $L_m(\mathrm{P}\Box\mathrm{S}\ ) = L_m(P) \cap L_m(S)$.

---

[1] Synchronous composition of two SM's can be defined in a more general setting, in which the event sets of the two SM's are different [2]. Our model can easily be generalized to address such cases.

Remark 2.3 We observe that $L(\text{P}\Box\text{S}) \subseteq \text{L(P)}$, and $L_m(\text{P}\Box\text{S}) \subseteq L_m(P)$.

Proposition 2.4 Let P denote the plant and S denote the supervisor. Then P□S is another supervisor for the plant P, and $\text{L}(\text{P}\Box(\text{P}\Box\text{S})) = \text{L}(\text{P}\Box\text{S})$, and $L_m(\text{P}\Box(\text{P}\Box\text{S})) = L_m(\text{P}\Box\text{S})$.

Let the event set Σ be partitioned into $\Sigma_u \cup \Sigma_c$, the set of *uncontrollable* and the *controllable* event sets.

Definition 2.5 A supervisor S is said to be *complete* [3] with respect to a given plant P if and only if for $s \in \Sigma^\star$ and $\sigma_u \in \Sigma_u$, the conditions $\gamma(s, z_0)!$ and $\alpha(s\sigma_u, x_0)!$ together imply that $\gamma(s\sigma_u, z_0)!$. In other words, the conditions $s \in \text{L}(\text{P}\Box\text{S})$ and $s\sigma_u \in \text{L(P)}$ together imply that $s\sigma_u \in \text{L}(\text{P}\Box\text{S})$.

A *controllable* language $\text{K} \subseteq \text{L}$ (with respect to L) satisfies the following invariance property: $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$.

Lemma 2.6 Let P denote the plant, S denote the supervisor and P□S denote the coupled system. Then L(P□S) is controllable with respect to L(P) iff the supervisor S is complete.

Proof: Assume that the supervisor S is complete; then the language generated by the coupled system satisfies the following invariance property (follows from Definition 2.5): $\text{L}(\text{P}\Box\text{S})\Sigma_u \cap L(P) \subseteq \text{L}(\text{P}\Box\text{S})$. Since the language generated by any SM is closed, we have $\text{L}(\text{P}\Box\text{S}) = \overline{L(P\Box S)}$. Thus we obtain, $\overline{L(P\Box S)}\ \Sigma_u \cap \text{L(P)} \subseteq \overline{L(P\Box S)}$. This means that L(P□S) is controllable with respect to L(P).

Assume now that L(P□S) is controllable with respect to L(P); now since L(P□S) is closed, we have the following: $\text{L}(\text{P}\Box\text{S})\Sigma_u \cap L(P) \subseteq L(P\Box S)$ This means that S is complete. Hence we get the desired result. Q. E. D.

Proposition 2.7 Let $K_1 \subseteq K_2 \subseteq \text{L(P)}$ be two given languages such that $K_1 \subseteq L_m(P) \subseteq \text{L(P)}$, and K_2 is prefix closed. Then there exists a supervisor S such that $L_m(\text{P} \Box \text{S}) = K_1$, and $\text{L}(\text{P} \Box \text{S}) = K_2$.

Proof: Let S_1 and S_2 be two supervisors for the plant P such that $L_m(S_1) = K_1$, $\text{L}(S_1) = \Sigma^\star$ and $L_m(S_2) = K_2$, $\text{L}(S_2) = K_2$; i. e. S_1 is the recognizer for K_1 and the S_2 is the generator for K_2. Since K_2 and $\Sigma^\star$ are closed, we can construct such SM's. All the states of S_2 are marked while S_1 can be viewed as a recognizer for K_1 with an additional dump state in order to generate $\Sigma^\star$. Let $\text{S} \overset{\text{def}}{=} S_1\Box S_2$ then

$$\begin{aligned} L_m(P\Box S) &= L_m(P\Box(S_1\Box S_2)) \\ &= L_m(P) \cap L_m(S_1) \cap L_m(S_2) \\ &= L_m(P) \cap K_1 \cap K_2 \\ &= K_1 \end{aligned}$$

Similarly,

$$\begin{aligned} \text{L}(\text{P}\Box\text{S}) &= \text{L(P)} \cap L(S_1) \cap L(S_2) \\ &= \text{L(P)} \cap \Sigma^\star \cap K_2 \\ &= K_2 \qquad \text{Q. E. D.} \end{aligned}$$

Remark 2.8 By Lemma 2.6, if the language K_2 is given to be controllable as well, then the supervisor S thus constructed will be complete. Combining the results of Lemma 2.6 and that of Proposition 2.7, we get the result stated in Proposition 5.1 in [3]. We note the simplification achieved in proving such a result due to the introduction of the notion of synchronous composition of two SM's in describing the control of DEDS's.

3. MINIMALLY RESTRICTIVE SUPERVISOR

In the previous section we have shown that given any plant P with uncontrolled behavior L(P), there exists a supervisor which when run synchronously with the plant can restrict its behavior to any closed subset K⊆L(P). We want the supervisor thus synthesized to be complete, so we would like to have the language K to be controllable with respect to L(P).

In case the prescribed language K is not given to be controllable with respect to L(P), we can construct a complete supervisor which when run synchronously with the plant generates the supremal controllable sublanguage of K. Let $K^{\dagger}$ denote the supremal controllable sublanguage K[2] of K. For the case when the languages L(P) and K are regular, an algorithm for constructing the language $K^{\dagger}$ is given in [4].

Definition 3.1 A complete supervisor S is said to be *minimally restrictive* if, when run synchronously with the plant P, it generates the supremal controllable sublanguage of the prescribed language.

Proposition 3.2 Let K $\subseteq$ L(P) be closed; then K is controllable with respect to L(P) iff $K\Sigma_u^{\star} \cap L(P) \subseteq K$, where $\Sigma_u^{\star}$ denotes the set of finite strings of uncontrollable events.

Proof: Assume that $K\Sigma_u^{\star} \cap L(P) \subseteq K$. Then $K\Sigma_u \cap L(P) \subseteq K$, since $\Sigma_u \subseteq \Sigma_u^{\star}$. Thus K is controllable, since K is closed . Given $K \subseteq L(P)$, $K\Sigma_u^{\star} \cap L(P) \subseteq K$ is always true for the zero length strings in $\Sigma_u^{\star}$.

Assume now that K is controllable . Hence, if $s \in K$, $\sigma_1 \in \Sigma_u$, and $s\sigma_1 \in L(P)$, then $s\sigma_1 \in K$. Similarly, if $(s\sigma_1) \in K$, $\sigma_2 \in \Sigma_u$, and $(s\sigma_1)\sigma_2 \in L(P)$, then $(s\sigma_1)\sigma_2 \in K$. Let Σ_u^2 denote the set of strings of uncontrollable events, of length two, then from the above two statements we obtain that, if $s \in K$, $(\sigma_1\sigma_2) \in \Sigma_u^2$, and $s(\sigma_1\sigma_2) \in L(P)(\Rightarrow s\sigma_1 \in L(P)$, since L(P) is closed), then $s(\sigma_1\sigma_2) \in K$ (and $s\sigma_1 \in K$). Continuing further in this manner we find; if $s \in K$, $u \in \Sigma_u^{\star}$, and $su \in L(P)(\Rightarrow s\overline{u} \in L(P)$, since L(P) is closed), then $su \in K$ (and $s\overline{u} \in K$). Thus $K\Sigma_u^{\star} \cap L(P) \subseteq K$. Q. E. D.

Corollary 3.3 K is controllable with respect to L(P) iff $\overline{K}\Sigma_u^{\star} \cap L(P) \subseteq \overline{K}$.

Given a deterministic SM, M $\equiv (Q, \Sigma, \delta, q_0, Q_m)$, there is a natural equivalence relation R_M [1], [7] induced by M on $\Sigma^{\star}$, which is defined by $s \equiv t(R_M) \Leftrightarrow \delta(s, q_0) = \delta(t, q_0)$[3], where $s, t \in \Sigma^{\star}$. We use $[s](R_m)$ to denote the equivalence class under the equivalence relation R_m, containing the string s.

Definition 3.4 A SM, M_1, is said to *refine* another SM, M_2, if for $s, t \in \Sigma^{\star}, s \equiv t(R_{M_1}) \Rightarrow s \equiv t(R_{M_2})$.

Lemma 3.5 Let $s, t \in$ L(P□S) be such that $s \equiv t(R_{P\Box S})$; then $s \equiv t(R_P)$, i. e. the SM, P□S, refines the SM, P.

Let P $\equiv (X, \Sigma, \alpha, x_0, X_m)$ denote the plant, S $\equiv (Y, \Sigma, \beta, y_0, Y_m)$ denote a supervisor that is not complete, and P□S$\equiv (Z, \Sigma, \gamma, z_0, Z_m)$ denote the coupled system. Let K = L(P□S) be the closed language generated by the coupled system; then by Lemma 2.6, K is not controllable. We want to construct a generator for $K^{\dagger}$, the supremal controllable sublanguage of K (Since K is closed, so is $K^{\dagger}$. Hence by Proposition 2.7, it is possible to construct a generator for $K^{\dagger}$).

Since K is not controllable, by Proposition 3.2, there exists $s \in K$ such that $su \in L(P)$ and $su \notin K$ for some $u \in \Sigma_u^{\star}$ (and conversely). We say that such a string $s \in K$

---

[2] Such supremal controllable language is well defined and is unique [4]. It is further shown in [4] that if the given languages K and L(P) are regular then the language $K^{\dagger}$ is also regular, i. e. can be generated by a FSM.

[3] This is meant to include the condition that $\delta(s, q_0)$ is undefined $\Leftrightarrow \delta(t, q_0)$ is undefined.

is an *uncontrollable* string. Let $s \in K$ be an uncontrollable string, and $z \in Z$ be such that $z(= (x_z, y_z))$, and $z = \gamma(s, z_0)$, i. e. z is the state reached by the SM, P$\Box$S, by accepting the string s. Then $x_z = \alpha(s, x_0)$ and $y_z = \beta(s, y_0)$. We now define; $\Sigma_u^\star(x_z) \overset{\text{def}}{=} \{u \in \Sigma_u^\star \mid \alpha(u, x_z)!\}$, and $\Sigma_u^\star(z) \overset{\text{def}}{=} \{u \in \Sigma_u^\star \mid \gamma(u, z)!\}$. It is clear from the previous discussion that s is uncontrollable iff $\Sigma_u^\star(x_z) \not\subseteq \Sigma_u^\star(z)$.

Proposition 3.6 Let $s \in K$ be an uncontrollable string. Then all the strings $t \in K$ such that $t \equiv s(R_{P\Box S})$ are uncontrollable.

Proof: $s \in K$ implies $s \in L(P)$. Let $z(= (x_z, y_z)) = \gamma(s, z_0)$ be the state reached by the SM, P$\Box$S, by accepting the string s; then $x_z = \alpha(s, x_0)$ is the state reached by the SM, P, by accepting the string s. Since s is uncontrollable, we have $\Sigma_u^\star(x_z) \not\subseteq \Sigma_u^\star(z)$.

Let t satisfy the hypothesis of the Proposition; then $t \equiv s(R_P)$, since by Lemma 3.5, P$\Box$S refines P. This means that the states reached by P$\Box$S and P by accepting the string t are again z and x_z respectively. But $\Sigma_u^\star(x_z) \not\subseteq \Sigma_u^\star(z)$, so t is uncontrollable. Q. E. D.

From the above Proposition it is clear that s is uncontrollable if and only if all the strings in the equivalence class $[s](R_{P\Box S})$ are uncontrollable. Thus we have that all the strings in the equivalence class $[s](R_{P\Box S})$ are uncontrollable if and only if $\Sigma_u^\star(x_z) \not\subseteq \Sigma_u^\star(z)$. We define, $Z_b \overset{\text{def}}{=} \{z \in Z \mid \Sigma_u^\star(x_z) \not\subseteq \Sigma_u^\star(z)\}$.

Remark 3.7 Thus, the deletion of the uncontrollable strings from K for obtaining $K^\uparrow$, is equivalent to the elimination of the states in the set Z_b from the SM realization of P$\Box$S.

Let $\Sigma_u(x_z) \overset{\text{def}}{=} \{\sigma_u \in \Sigma_u \mid \alpha(\sigma_u, x_z)!\}$, and $\Sigma_u(z) \overset{\text{def}}{=} \{\sigma_u \in \Sigma_u \mid \gamma(\sigma_u, z)!\}$. We partition Z_b into two sets $Z_b^1 \cup Z_b^2$, defined below:

$Z_b^1 \overset{\text{def}}{=} \{z \in Z \mid \Sigma_u(x_z) \not\subseteq \Sigma_u(z)\}$, and $Z_b^2 \overset{\text{def}}{=} \{z \in Z - Z_b^1 \mid \gamma(u, z) \in Z_b^1 \text{ for some } u \in \Sigma_u^\star\}$.

Proposition 3.8 Let Z_b, Z_b^1, and Z_b^2 be as defined above; then $Z_b = Z_b^1 \cup Z_b^2$.

Proof: We shall first show that $Z_b^1 \cup Z_b^2 \subseteq Z_b$. We have $Z_b^1 \subseteq Z_b$, since $\Sigma_u \subseteq \Sigma_u^\star$. To show that $Z_b^2 \subseteq Z_b$; let $\tilde{z} \in Z_b^2$, then there exists $z \in Z_b^1$ such that $\gamma(u, \tilde{z}) = z$ for some $u \in \Sigma_u^\star$. Since $z \in Z_b^1$, there exists $\sigma_u \in \Sigma_u$ such that $\sigma_u \in \Sigma_u(x_z)$ and $\sigma_u \notin \Sigma_u(z)$. Thus, $u\sigma_u \in \Sigma_u^\star$ (since $u \in \Sigma_u^\star$ and $\sigma_u \in \Sigma_u$), $u\sigma_u \in \Sigma_u^\star(x_{\tilde{z}})$, and $u\sigma_u \notin \Sigma_u^\star(\tilde{z})$. This implies that $\Sigma_u^\star(x_{\tilde{z}}) \not\subseteq \Sigma_u^\star(\tilde{z})$, so $\tilde{z} \in Z_b$. Thus $Z_b^2 \subseteq Z_b$. Hence we obtain, $Z_b^1 \cup Z_b^2 \subseteq Z_b$.

Next we shall show that $Z_b \subseteq Z_b^1 \cup Z_b^2$. Let $z \in Z_b$; then there exists $u \in \Sigma_u^\star$ such that $u \in \Sigma_u^\star(x_z)$, and $u \notin \Sigma_u^\star(z)$. Let u be the smallest such string; then either $u = \sigma_u$ for some $\sigma_u \in \Sigma_u$, in which case $\Sigma_u(x_z) \not\subseteq \Sigma_u(z)$, i. e. $z \in Z_b^1$ or $u = v\sigma_u$ for some $v \in \Sigma_u^\star$ and $\sigma_u \in \Sigma_u$, in which case $\gamma(v, z) \in Z_b^1$, i. e. $z \in Z_b^2$. Thus, if $z \in Z_b$, then either $z \in Z_b^1$ or $z \in Z_b^2$. This implies that $z \in Z_b^1 \cup Z_b^2$. Thus, $Z_b \subseteq Z_b^1 \cup Z_b^2$. Q. E. D.

Remark 3.9 Thus, in order to construct the generator for $K^\uparrow$, we need to eliminate the states in the set Z_b^1 and those in Z_b^2 from the SM realization of P$\Box$S.

Algorithm: We assume that the all the SM's have finite[4] states, i. e. , L(P) and K are regular languages.

1. Construct the FSM, P$\Box$S.
2. Determine the states in the set $Z_b^1 = \{z \in Z \mid \Sigma_u(x_z) \not\subseteq \Sigma_u(z)\}$.
3. Construct the FSM, P$\Box$S$|_{\Sigma_u} \equiv (Z, \Sigma_u, \gamma|_{\Sigma_u \times Z}, z_0, Z_m)$ by deleting all the transitions caused by the controllable events.
4. Determine the states in the set Z_b^2 by picking those states $z \in Z - Z_b^1$ which have a path of transitions leading to some state in Z_b^1 in the FSM, P$\Box$S$|_{\Sigma_u}$.

---

[4] Finiteness of the number of the states is required for the algorithm to terminate in finite time.

5. Delete the states $Z_b^1 \cup Z_b^2$ from the FSM realization of P□S to obtain the generator for $K^\dagger$.

Computational Complexity: Let m and n denote the number of states in the minimal FSM realizations of the languages L(P) and K respectively.

1. The FSM, P□S can be constructed in order $O(mn)$ time, since the FSM, P□S can have a maximum number mn of states.
2. The states in the set Z_b^1 can be identified in order $O(mn)$ time, because the maximum number of possible bad states is mn.
3. The FSM P□S$|_{\Sigma_u}$ can again be constructed in order $O(mn)$ time, because in order to delete the transitions caused by the controllable events, we need to consider the transitions in all the possible mn states.
4. The states in the set Z_b^2 can be identified in order $O(mn)$ time again, because, if we consider the FSM P□S$|_{\Sigma_u}$ with all its transitions reversed, then the states in the set Z_b^2 are those states that are *reachable* from the states in the set Z_b^1.
   Proof: Assume, for example, that the number of states in the set Z_b^1 is $p(\leq mn)$. Then the states that can be reached by a single transition from these p states in the FSM, P□S$|_{\Sigma_u}$ with all its transitions reversed, can be identified in order $O(p)$ time. Let the number of such states be $q(\leq (mn-p))$. Then the states that can be reached from these q states in the FSM, P□S$|_{\Sigma_u}$ with all its transitions reversed, due to a single transition, can be identified in order $O(q)$ time. Thus the states reachable from the states in the set Z_b^1 in the FSM, P□S$|_{\Sigma_u}$ with all its transitions reversed, due to two successive transitions, can be identified in order $O(p+q)(\leq O(mn))$ time. If we continue identifying the states that are reachable from the states in the set Z_b^1 in the above fashion, then it is easy to see that all such states can be identified in order $\leq O(mn)$ time. This justifies our last claim (While determining the states reachable from the set Z_b^1 care is taken so that none of the states is explored more than once). Q. E. D.

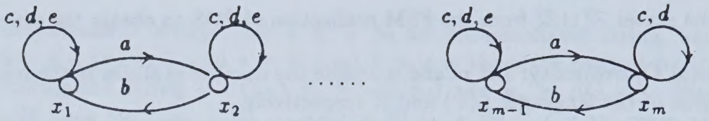
Remark 3.10 Thus the computational complexity of the above algorithm for constructing the generator of the language $K^\dagger$ is of order $O(mn)$, where m and n are the number of states in the minimal FSM realizations of the languages L(P) and K respectively [5]. Our algorithm for constructing the generator for $K^\dagger$ is an order of magnitude faster than the one in [4]. The computational complexity of the algorithm in [4] is of order $O(m^2n^2)$ [6].

Optimality of the Algorithm: We shall show now that the above algorithm for constructing the supremal controllable sublanguage is also an *optimal* algorithm, i. e. given the plant P that generates the language L(P) and another closed language K with m and n states in their minimal FSM realizations, respectively, the supremal controllable sublanguage of K with respect to L(P) cannot in general be computed faster than $O(mn)$ time. We provide an example of languages L(P) and K with m and n states in their minimal FSM realizations such that the generator for $K^\dagger$ has $O(mn)$ states in its FSM realization, thus showing that any algorithm for constructing the generator for $K^\dagger$ must be of computational complexity $O(mn)$.

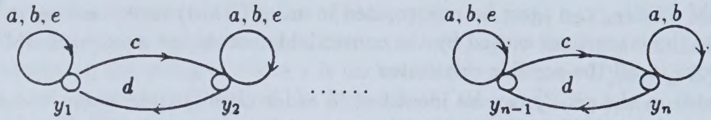
Let $\Sigma = \{a, b, c, d, e\}$ and $\Sigma_u = \{e\}$. Let L(P) and K be as given below: $L(P) = \{s \in \Sigma^\star \mid 0 \leq \#(a,s) - \#(b,s) \leq (m-1)$ and for all prefixes we of s, $0 \leq \#(a,w) - \#(b,w) \leq (m-2)\}$ and $K = \{s \in \Sigma^\star \mid 0 \leq \#(c,s) - \#(d,s) \leq (n-1)$ and for all prefixes we of s, $0 \leq \#(c,w) - \#(d,w) \leq (n-2)\}$; where the notation $\#(x,y)$ denotes the number of events "x" in the string "y".

---

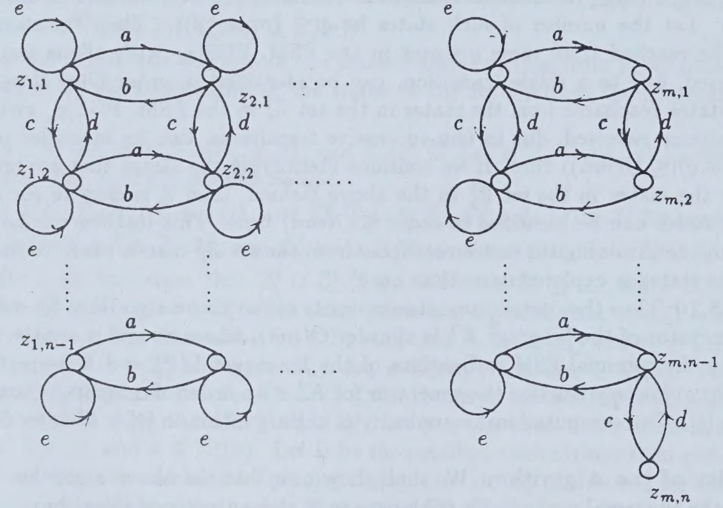
[5] We have neglected the dependence of the computational complexity on any parameter other than m and n.


c, d, e
c, d, e
c, d, e
c, d
a
b
a
b
x_1
x_2
x_{m-1}
x_m
……

Generator for the language L(P) (m states)


a, b, e
a, b, e
a, b, e
a, b
c
d
c
d
y_1
y_2
y_{n-1}
y_n
……

Generator for the language K (n states)


e
a
b
c
d
$z_{1,1}$
$z_{2,1}$
$z_{m,1}$
$z_{1,2}$
$z_{2,2}$
$z_{m,2}$
$z_{1,n-1}$
$z_{m,n-1}$
$z_{m,n}$
……

Generator for the language $K^{\uparrow}$ (m(n-1)+1 states)

Figure 1: Example to illustrate optimality of the algorithm

The language K is not controllable with respect to L(P), for the string $c^{n-1}e \in$ L(P) but $c^{n-1}e \notin$ K. If we delete all such uncontrollable strings from K we get the the following language as $K^{\uparrow}$: $K^{\uparrow} = \{s \in \Sigma^{\star} \mid 0 \leq \#(a,s) - \#(b,s) \leq (m-2)$ and $0 \leq \#(c,s) - \#(d,s) \leq (n-2)\} \cup \{s \in \{a,b,c,d\}^{\star} \mid 0 \leq \#(a,s) - \#(b,s) = (m-1)$ and $0 \leq \#(c,s) - \#(d,s) \leq (n-1)\}$. The generators of languages L(P), K and $K^{\uparrow}$ are shown in Figure 1.

In the above example L(P) and K have m and n states, respectively, in their minimal FSM realizations, whereas $K^{\uparrow}$ has $m(n-1)+1$ states in its minimal FSM realization; thus any algorithm that computes $K^{\uparrow}$ must take $O(mn)$ time.

3.1. CLOSED FORM REPRESENTATION OF $K^{\uparrow}$: We have given an optimal algorithm for constructing the supremal controllable sublanguage for the case when the languages L(P) and K are closed and regular. We now present a closed form representation of the language $K^{\uparrow}$ for the case when L(P) and K are assumed to be closed.

Definition 3.11 Given two languages L_1 and L_2 the *quotient* L_1/L_2 is defined to be $\{s \mid \exists t \in L_2 \text{ such that } st \in L_1\}$.

Proposition 3.12 Let K and L(P) be prefix closed languges. Then $K^{\uparrow} = K - ((L(P) - K)/\Sigma_u^*)\Sigma^*$.

Proof: Let s be an uncontrollable string in K. Then by Proposition 3.2 there exists $u \in \Sigma_u^*$ such that $su \in L(P)$ and $su \notin K$, i. e. $su \in (L(P) - K)$, i. e. $s \in (L(P) - K)/\Sigma_u^*$. Now, if $s \in$ K is an uncontrollable string then all the strings in K of which s is a prefix must also be deleted from K in order to construct $K^{\uparrow}$ (this follows from the definition of a controllable language). Thus the set of strings that must be deleted from K is the set $((L(P) - K)/\Sigma_u^*)\Sigma^*$. This implies that $K^{\uparrow} = K - ((L(P) - K)/\Sigma_u^*)\Sigma^*$. Q. E. D.

Remark 3.13 From the above closed form representation of $K^{\uparrow}$ it is easy to see that $K^{\uparrow}$ is closed. Also, if L(P) and K are given to be regular, then $K^{\uparrow}$ is regular as well. If we employed the above closed form expression of $K^{\uparrow}$ for developing an algorithm for constructing the generator for $K^{\uparrow}$, the computational complexity of the algorithm would be of order $O(mn^2)$ which is clearly not optimal.

3.2. ALGORITHM FOR CONSTRUCTING $K^{\uparrow}$ (GENERAL CASE): Consider two languages L and K both regular but not necessarily closed, $K \subseteq L$. Our objective is to construct the recognizer for the supremal controllable sublanguage of K with respect to L. The algorithm is similar to the one for the case when both K and L are given to be closed and regular.

Since K and L are not closed they cannot be *generated* by FSM's. Let P and S be *recognizers* for L and K respectively, i. e. $L_m(P) = L$ and $L_m(S) = K$, such that $\overline{L_m(P)} = L(P)$ and $\overline{L_m(S)} = L(S)$. As before, P denotes the plant and S the supervisor. Let P$\Box$S denote the coupled system, then by Lemma 2.2 $L_m(P\Box S) = K$. Since K is not controllable we have $\overline{K}\Sigma_u \cap L_m(P) \not\subseteq \overline{K}$, i. e. there exists an uncontrollable string $s \in \overline{K}$ and $\sigma_u \in \Sigma_u$ such that $s\sigma_u \in L_m(P)$ and $s\sigma_u \notin \overline{K}$. As before, given the uncontrollable string $s \in K$, all strings in the equivalence class $[s](R_{P\Box S})$ are uncontrollable. We must now delete all states that are reached by accepting the uncontrollable strings from the FSM, P$\Box$S in order to get the recognizer for$K^{\uparrow}$. Let $z(= (x_z, y_z))$ be the state reached in the FSM P$\Box$S by accepting the uncontrollable string s; then there exists $\sigma_u \in \Sigma_u$ such that the following two conditions are satisfied;

1. $\alpha(\sigma_u, x_z) \in X_m$,i. e. $s\sigma_u \in L_m(P)$, and
2. $\gamma(\sigma_u, z)$ is not defined; i. e. $\Sigma_u(x_z) \not\subseteq \Sigma_u(z)$.

Let the set of all states in P$\Box$S reached by accepting uncontrollble strings be denoted by the set Z_b^1. If we delete the states in the set Z_b^1 from the FSM P$\Box$S, some more states might satisfy conditions 1 and 2 due to the removal of the states and the transitions leading to them. We can identify such states simply by considering the FSM, P$\Box S \mid_{\Sigma_u \times X_m}$, where

$$\text{P}\Box S \mid_{\Sigma_u \times X_m} \equiv ((X_m, Y), \Sigma_u, \gamma \mid_{\Sigma_u \times (X_m, Y)}, z_0, Z_m)$$

Let Z_b^2 denote the collection of states that satisfy conditions 1 and 2 due to the deletion of the states in the set Z_b^1, i. e. $Z_b^2 = \{z \in Z - Z_b^1 \mid \exists u \in \Sigma_u^* \text{ such that } \gamma \mid_{\Sigma_u \times (X_m, Y)} (u, z) \in Z_b^1\}$ From the above discussions the following proposition is immediate.

Proposition 3.14 Let $Z_b = Z_b^1 \cup Z_b^2$; then Z_b is the set of all the states that must be deleted from the FSM P$\Box$S to obtain the recognizer for $K^{\uparrow}$.

Algorithm: The algorithm for constructing the recognizer for P□S is very similar to the one in which the languages K and L are closed and regular and basically consists of constructing the coupled system P□S and deleting the states in the set Z_b from it. A similar analysis as before shows that the algorithm for constructing $K^\uparrow$ is linear in the product of the number of states in the minimal FSM realizations of L and K. The problem of constructing $K^\uparrow$ when K and L are not closed is computationally harder than the one in which they are closed, but still we compute $K^\uparrow$ in $O(mn)$ time, thus showing that the above algorithm is optimal.

4. ON SUPREMAL OBSERVABLE LANGUAGES

In all our previous discussions we have assumed that the supervisor has complete information about the plant behavior, i. e. the supervisor observes all the events that occur in the plant perfectly. However, in many situations the supervisor has only partial information about the plant behavior. Let M : $\Sigma \rightarrow \Delta$ be a map, called a *mask*, and Δ be the set of events that are observed by the supervisor. Let P denote the plant and K$\subseteq L(P)$ denote the desired behavior. In order to be able to construct a supervisor so that the coupled system has the desired behavior K, the language K must be *observable* with respect to L(P).

Definition 4.1 A language K is said to be observable with respect to another language L if and only if given $s \in L \cap K$, $t \in L$, M(t) = M(s) implies $t \in K$.

If the given language K is not observable we construct a supervisor so that the coupled system generates the supremal observable sublanguage of K (The supremal observable sublanguage exists and is unique [9]). For the treatment of observability of any given language K with respect to L we need only consider the subset of K contained in L. Hence we will assume that K is a subset of L.

Proposition 4.2 The supremal observable sublanguage of K with respect to L is $K - M^{-1}M(L-K)$.

Proof: If K is observable, then given $s \in K$ there does not exist $t \in L - K$ such that M(t) = M(s); also given $s \in (L-K)$ there does not exist $t \in K$ such that M(t) = M(s). Thus $M^{-1}M(L-K) = (L-K)$. The last equality implies that $K - M^{-1}M(L-K) = K - (L-K) = K$.

If K is not observable then there exists $s \in K$ and $t \in L$ such that M(t) = M(s) and $t \notin K$, i. e. there exists $t \in (L-K)$ such that M(t) = M(s). Such a string $s \in K$ is called an *unobservable* string. We must delete all such strings from K to get its supremal observable sublanguage. Now, if $s \in K$ and $t \in L - K$ such that M(t) = M(s), then $s \in M^{-1}M(L-K)$. Thus the set $M^{-1}M(L-K)$ consists of all the unobservable strings of K, and hence the set $K - M^{-1}M(L-K)$ is the supremal observable sublanguage of K. Q. E. D.

Remark 4.3 We can use this closed form representation to construct an algorithm for computing the supremal observable sublanguage of K with respect to L, the computational complexity of which would be exponential in the product of the number of states in the minimal FSM realizations of K and L, as expected [8].

Corollary 4.4 If $K \subseteq L$ is closed, then the supremal closed and observable sublanguage of K with respect to L is the set $K - M^{-1}M(L-K)\Sigma^\star$.

Proof: In order to obtain the supremal closed and observable sublanguage of K, we need to delete the set containing all of the unobservable strigs and their extensions (in K) from K. The set $M^{-1}M(L-K)\Sigma^\star$ contains all the unobservable strings of K (see the proof above) as well as their extensions in $\Sigma^\star$. Hence the set $K - M^{-1}M(L-K)\Sigma^\star$ is the required supremal closed and observable subset of K. Q. E. D.

5. CONCLUSION

In this paper, we have used the notion of *synchronous composition* of two SM's for describing the control of DEDS's. This considerably simplifies the problem of synthesizing supervisors for controlling any uncontrolled plant.

We have defined the notion of a complete supervisor in the context of synchronous composition of plants and supervisors and shown that the language generated by the synchronous composition of a plant and a complete supervisor is controllable, and conversely. An equivalent represention of controllable languages has been developed, and been used to design an algorithm for constructing the minimally restrictive supervisor. The computational complexity of this algorithm is linear in the product of the number of states in the FSM realizations of the language generated by the plant and the prescribed language that is not controllable. Computational optimality of the above algorithm has been proved.

We have modified the above algorithm to construct the supremal controllable sublanguage for the case when the languages are not closed and shown it to be computationally optimal. Closed form representations of the supremal controllable (for the case when the languages are closed) and the supremal observable languages (for the general case) have also been presented.

6. REFERENCES

1. Hopcroft, J. E. and J. D. Ullman, *"Introduction to Automata Theory, Languages and Computation,"* Addison–Wesley, 1979.
2. Hoare, C. A. R. , *"Communicating Sequential Processes,"* Englewood Cliffs, 1985.
3. Ramadge, P. J. and W. M. Wonham, *"Supervisory Control of a Class of Discrete Event Procesess,"* SIAM J. Control and Optim. **25** (1987) 206–230.
4. Ramadge, P. J. and W. M. Wonham, *"On the Supremal Controllable Sublanguage of a given Language,"* SIAM J. Control and Optim. **25** (1987) 637–659.
5. Garg, V. K. , *"Specification and Analysis of Distributed Systems with a large number of Processes,"* Ph. D. Theisis, Univ. of Cal., Berkeley, 1988.
6. Ramadge, P. J. , *"The Complexity of some Basic Problems for Discrete Event Systems,"* Advanced Computing Concepts and Techniques in Control Engr. NATO ASI Series, **F47**, Denham, M. J. and A. J. Laub, eds. , Springer-Verlag, 171– 190.
7. Cho, H. and S. I. Marcus, *"Supremal and Maximal Sublanguages Arising in Supervisor Synthesis Problems with Partial Observations,"* to appear in Mathematical Systems Theory.
8. Cho, H. and S. I. Marcus, *"On Supremal Languages of class of Sublanguages that Arise in Supervisor Synthesis Problems with Partial Observations,"* Math. Control Signals Systems, **2** (1989), 47-69.
9. Lin, F. and W. M. Wonham, *"On Observability of Discrete-Event Systems,"* Univ. Toronto, Toronto, Ont., Canada, Syst. Contr. Group Rep. 8701, Jan. 1987.
10. Kumar, R. , Garg, V. and Marcus, S. I. , *"On Supremal Controllable and Observable Sublanguages of Discerte Event Dynamical Sysytem,"* Technical Report, The Univ. of Texas at Austin, Austin, Texas 78712

HIERARCHICAL CONTROL OF DISCRETE EVENT SYSTEMS: COMPUTATION AND EXAMPLES

H. Zhong W.M. Wonham
Systems Control Group
Department of Electrical Engineering
University of Toronto
Toronto, Ontario, Canada
M5S 1A4

September 27, 1989

Abstract

The paper continues previous work of the authors on a hierarchical version of the control theory for discrete-event systems initiated by Ramadge and Wonham. Computational procedures are summarized for achieving hierarchical consistency between *operator* amd *managerial* levels, and the approach is illustrated by three simple applications.

1 Introduction

In this paper, we present outlines of computational procedures and three examples taken from potential application domains, continuing the discussion of hierarchical control initiated in [ZW88]. The general setup is illustrated by the two-level hierarchy of Fig. 1, consisting of a low-level plant G_{lo}, with its low-level controller C_{lo} (the *operator*), along with the high-level plant model G_{hi} and its associated controller C_{hi} (the *manager*). The link Inf_{lohi} is an information channel by which G_{hi} is updated, from time to time, following the occurrence of *significant* events in G_{lo}; while the link Com_{hilo} is a command channel by which instructions are transmitted by the manager to the operator. The links Con_{lo}, Inf_{lo} and Con_{hi}, Inf_{hi} represent conventional actuational and feedback channels at the operator and managerial levels respectively. However, we recall that actuation via Con_{hi} is really only *virtual*, in that actual control of the behavior of G_{hi} is mediated through the hierarchical control loop, namely through *command & control* via Com_{hilo}; Con_{lo}, followed by *report & advise* via Inf_{lohi}; Inf_{hi}.

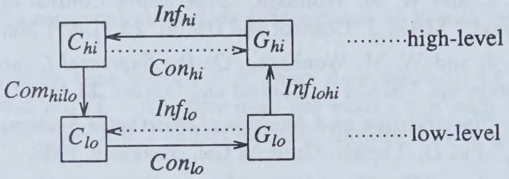


Fig. 1. Hierarchical structure

We analyzed the foregoing general setup in [ZW88] when both levels of the hierarchy are modelled as controlled discrete-event systems (DES) in the framework of Ramadge-Wonham [RW87]. In particular, the property that G_{lo} be *output control consistent (OCC)* was shown to be crucial in defining a meaningful control structure in G_{hi}, while the stronger property that G_{lo} be *strictly output control consistent (SOCC)* was shown to guarantee the ability of the hierarchical control loop to synthesize controllable sublanguages generable by G_{hi}. We can achieve these key properties by refining the structure of G_{lo} through conceptual modification of the tree grammar of $L(G_{lo})$. In this paper, we shall sketch computational algorithms that implement the procedures effectively (with polynomial time complexity) when the state and transition sets of G_{lo} are finite–the regular case. Applications are provided.

Recall from [ZW88] that the plant model is a mild extension of that in [RW87] to accommodate state outputs:

$$G_{lo} = (\Sigma, Q, T_0, \delta, \omega, q_0, Q_m)$$

Here Σ is the (finite) set of event labels, partitioned as usual into the *controllable* subset Σ_c and *uncontrollable* subset Σ_u; $T_0 = T \cup \{\tau_0\}$ is the state-output alphabet T augmented by a *silent* symbol τ_0; $\delta : \Sigma \times Q \rightarrow Q$ is the state-transition function (a partial function); $\omega : Q \rightarrow T_0$ is the state-output function; q_0 is the initial state (with $\omega(q_0) = \tau_0$); and $Q_m \subseteq Q$ is the subset of *marker* states, which we specialize here to be $Q_m = Q$. As usual, δ is extended by iteration to a partial function on $\Sigma^* \times Q$, the *closed behavior* $L(G_{lo})$ of G_{lo} being the set of all finite strings $s \in \Sigma^*$ for which the extension $\delta(s, q_0)$ is defined. Thus $L(G_{lo})$ is the *freest possible* behavior of G_{lo}, corresponding to a situation in which the controllable events (transition with label $\sigma \in \Sigma_c$) are all permanently enabled.

Let ϵ denote the empty string over any alphabet. It is clear that G_{lo} uniquely determines a map $\theta : L(G_{lo}) \rightarrow T^*$ according to

$$\begin{aligned} \theta(\epsilon) &= \epsilon \\ \theta(s\sigma) &= \begin{cases} \theta(s) & \text{if } \omega(\delta(s\sigma, q_0)) = \tau_0 \\ \theta(s)\tau & \text{if } \omega(\delta(s\sigma, q_0)) = \tau \text{ and } \tau \in T \end{cases} \end{aligned}$$

The map θ formalizes the reporting of information via Inf_{lohi}. The manager's model G_{hi} is taken to be any (say, a minimal-state) recognizer for the image of θ, namely

$$L(G_{hi}) = \theta(L(G_{lo})).$$

When G_{lo} is OCC, the alphabet T can be meaningfully partitioned as

$$T = T_c \cup T_u$$

in the same sense as Σ. This means that, following the appearance of a string $t \in T^*$ in Inf_{hi}, an event (labelled) $\tau \in T_c$ can always be disabled via command & control, namely the manager can disable controllable events in G_{hi} (via Com_{hilo}) just as if he had direct control over G_{hi} via the virtual actuation channel Con_{hi} (see [ZW88] for details). Suppose that a G_{hi}-controllable sublanguage (defined in [RW87]) is specified. The manager can exercise command & control by suitable disablements of events in T_c with the aim of synthesizing E_{hi}. When G_{lo} is merely OCC, the behavior thereby induced in G_{hi} (via Inf_{lohi}) is in general only a proper sublanguage of E_{hi}. The reason is that disablement of an event τ_c may entail as an undesired side effect (unbeknownst to C_{hi}) the disablement of other controllable events $\tau_{c'}$. However, when G_{lo} is strictly OCC (*i.e.* SOCC) equality is achieved for arbitrary controllable $E_{hi} \subseteq L(G_{hi})$, and then we say that the pair (G_{lo}, G_{hi}) is *hierarchically consistent.*

2 Algorithmic Structural Design for Hierarchical Consistency

This section is closely keyed to [ZW88]. Starting with an arbitrary structure G_{lo} as described above, our first objective is to create a modified structure $G_{lo,occ}$ such that

$$L(G_{lo,occ}) = L(G_{lo})$$

and $G_{lo,occ}$ is OCC. In addition, information provided by the original updating map must be preserved, namely $\theta : \Sigma^* \rightarrow T^*$ is required to factor through $\theta_{occ} : \Sigma^* \rightarrow T^*_{occ}$. For this, we convert the conceptual procedure of [ZW88] to an algorithm that modifies the (finite) transition structure of G_{lo} as follows. In Q a state q is *silent* if $\omega(q) = \tau_0$ or *vocal* if $\omega(q) = \tau \in T$. A path in G_{lo}, displayed as

$$q' \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \longrightarrow \cdots \xrightarrow{\sigma_{k-1}} q_{k-1} \xrightarrow{\sigma_k} q$$

is *silent* if (i) q' is either a vocal state or the initial state, and (ii) either $k = 1$, or $k > 1$ and the states $q_1, \cdots, q_{k-1}$ are silent. A silent path is *red* if at least one of its transition labels $\sigma_1, \cdots, \sigma_k$ is controllable; otherwise it is *green*. For $q \in Q$ let $P(q)$ be the set of all silent paths that end at q. Say that $q \in Q$ is *color consistent* if either every path in $P(q)$ is red (then q is colored *red*) or every path in $P(q)$ is green (and q is colored *green*). The desired property OCC is that each $q \in Q$ be color consistent.

In general, for G_{lo} there will be states $q \in Q$ for which some paths in $P(q)$ are red and some are green (q is then colored *amber*): denote these path subsets by $P_r(q)$, $P_g(q)$. At an amber state q do the following: (i) bring in two new states (*siblings*) q_r, q_g colored red and green respectively: if q is silent, so are its siblings, but if q is vocal with output τ, then make q_r vocal with output τ_c (the *controllable instance* of τ) while similarly make q_g vocal with output τ_u; (ii) for each transition $[q, \sigma, q']$ bring in two new transitions $[q_r, \sigma, q']$, $[q_g, \sigma, q']$; (iii) change the terminal state of every path in $P_r(q)$ from q to q_r, and of every path in $P_g(q)$ from q to q_g; (iv) delete q and its associated transitions from the structure. The result is a new structure, say $G_{lo}^{(1)}$, with the same behavior, *i.e.*

$$L(G_{lo}^{(1)}) = L(G_{lo})$$

but with one less color inconsistent state; and by the new output coding, state-output information has been preserved. The algorithm is initialized by coloring each state green; then amber states are repeatedly identified and processed as described (except that no state in G_{lo} is split more than once). The final result $G_{lo,occ}$ is OCC, and has state size $|Q_{occ}|$ and output alphabet size $|T_{occ}|$ at most double their counterparts in G_{lo}. Let G_{lo} have n states and m transitions. The authors' Prolog implementation of the algorithm (called POCC) runs in time $O(n^2m + nm^2)$.

To obtain hierarchical consistency, we adapt the conceptual procedure [ZW88] that achieves the SOCC property in the tree grammar, except that the result we achieve for the finite transition structure is somewhat weaker, though sufficient for the purpose. Start from the model $G_{lo,occ}$ which we now denote simply by G_{lo}. Each state $q \in Q$ is now unambiguously red or green. A *silent path suffix (sps)* in G_{lo} is a path as displayed above, with $q_1, \cdots, q_{k-1}$ silent and q vocal, but q' unrestricted. An *agent* is a pair $(q', \sigma) \in Q \times \Sigma_c$ such that there is an sps as displayed with $\sigma_1 = \sigma \in \Sigma_c$ and $\sigma_2, \cdots, \sigma_k \in \Sigma_u$ (thus q is red). Say q is *critical* for (q', σ), and denote the subset of critical states for (q', σ) by $C(q', \sigma)$. Suppose $q'' \in C(q', \sigma)$ and let $\tau'' = \omega(q'')$; thus $\tau'' \in T_c$. By command & control [ZW88], σ will be disabled by Con_{lo} at state q' if τ'' is currently disabled by Con_{hi}. Next define the *blocking set* $B(q', \sigma)$ of an agent (q', σ) to be the subset of (red, green) states $q \in Q$ such that an sps exists as displayed, with $\sigma_1 = \sigma$ but now with $\sigma_2, \cdots, \sigma_k$ unrestricted; thus $C(q', \sigma) \subseteq B(q', \sigma)$. An agent (q', σ) is *unary* if $|B(q', \sigma)| = 1$, *i.e.* contains just one state, which is thus critical. Let $p, q \in Q$ with p red and vocal but q arbitrary. An sps from q to p is *dedicated* if each of its transitions $[p_1, \gamma, p_2]$ is such that either (p_1, γ) is not an agent, or (p_1, γ) is a unary agent with $B(p_1, \gamma) = \{p\}$. Define the set

$$D(q) = \{p \in Q | p \text{ is red and vocal \& there is a dedicated sps joining } q \text{ to } p\}$$

An agent (q', σ) is *admissible* if

$$B(q', \sigma) \subseteq D(q');$$

that is, each state potentially blocked by (q', σ) is reachable from q' along a dedicated sps. Otherwise (q', σ) is *inadmissible*. The notion of inadmissible agent plays the role of the *antecedent* of *partners* in [ZW88]: it can be shown that if an inadmissible agent is present in the (finite) transition structure of G_{lo}, then partners are present in the tree grammar of $L(G_{lo})$, *i.e.* G_{lo} fails to be SOCC. While the converse statement is false, we can show nevertheless that if G_{lo} is OCC and is free of inadmissible agents, (G_{lo}, G_{hi}) is hierarchically consistent.

Our second algorithm (called PHCC in the Prolog implementation) identifies all inadmissible agents (q', σ), vocalizes their entrance state $\delta(\sigma, q')$ by attaching a new state-output symbol, say τ_{new} (the same in each case), and redefining ω accordingly. Unfortunately, in this process the property of OCC may be destroyed; therefore, it is necessary to subject the result to POCC once more. Happily this step cannot create inadmissible agents, and so the final result $G_{lo,new}$ has the properties required. The new updating map θ_{new} is defined in the same way as previously for θ. Clearly θ factors through θ_{new} (simply erase τ_{new}, or $\tau_{new,c}$ and $\tau_{new,u}$ as appropriate from $\theta_{new}(s)$), so updating information is preserved. Just as for G_{hi}, the manager's new model $G_{hi,new}$ is defined to be a transition structure that generates exactly $\theta_{new}(L(G_{lo,new}))$. Thus the pair $(G_{lo,new}, G_{hi,new})$ is hierarchically consistent, as we set out to achieve.

Detailed analysis shows that PHCC runs in $O(n^3 + n^2m)$ time, so that the required sequence POCC;PHCC;POCC runs in $O(n^3 + n^2m + nm^2)$ time, not very cheap but in principle tractable. The worst-case enlargement of the state set over that for G_{lo} as originally given is a factor of 4; but in most realistic examples a factor around 1.5 seems to be typical. The size of the manager's alphabet T will in the worst case increase from $|T|$ to $2(|T| + 1)$.

3 Examples

3.1 Transfer Line

We consider a transfer line (Fig. 2) consisting of two machines M_1, M_2, two buffers B_1, B_2,

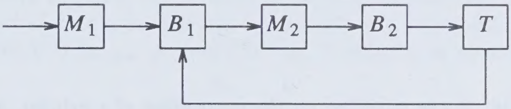


Fig. 2. Example of a transfer line

and a test unit T (cf. [AJ88]). A workpiece is processed by M_1 followed by M_2, then tested by T. If the test is passed, the workpiece exits; otherwise it is returned to B_1 for reprocessing by M_2. Each buffer has capacity 1.

The state diagram TL_{lo} is displayed in Fig. 3. We have

$$\Sigma_c = \{l\}, \quad \Sigma_u = \{a, r, t\}$$

where the event labels stand for *load*, *advance*, *return*, and *start test*. The significant events

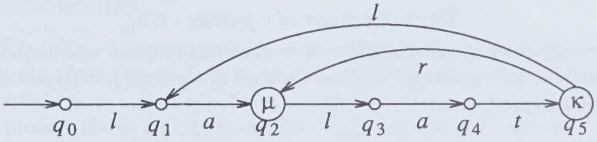


Fig. 3. TL_{lo}

reported to the manager are μ (workpiece advances to B_1) and κ (start test, *i.e.* workpiece enters T). Notice that TL_{lo} is not OCC because μ is not unambiguously controllable or uncontrollable: μ can be disabled from initial state q_0 but not from the vocal state q_5. The ambiguity is resolved by application to TL_{lo} of POCC, with the result shown in Fig.4. In the new OCC structure $TL_{lo,occ}$, the (old) state q_2 has been split into new states q_2, q_6 with outputs the controllable and uncontrollable instances of μ, labelled μ_c and μ_u. By inspection, $TL_{lo,occ}$ is free of inadmissible agents, and this can be verified by application of PHCC, so we set $TL_{lo,new} = TL_{lo,occ}$. The manager's model $TL_{hi,new}$ (Fig. 5) can be used for command & control with

$$T_c = \{\kappa, \mu_c\}, \quad T_u = \{\mu_u\}.$$

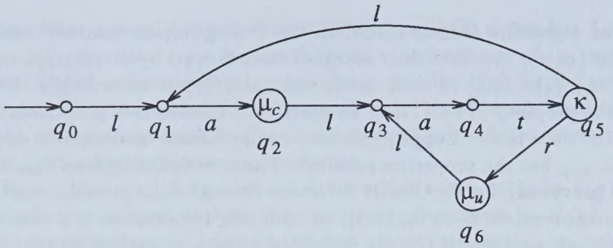


Fig. 4. $TL_{lo,occ}$

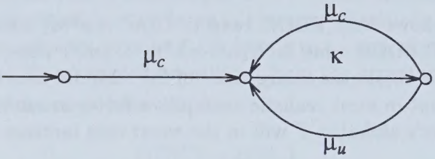


Fig. 5. $TL_{hi,new}$

For instance, $TL_{hi,new}$ (unlike TL_{hi}) allows a record to be kept of the frequency of test failures (μ_u), and thus the manager may shut down the transfer line if the count of events μ_u becomes unacceptable.

3.2 Grinder

The DES GR_{lo} of Fig. 6 was modelled on the description of a grinder, but is more-or-less

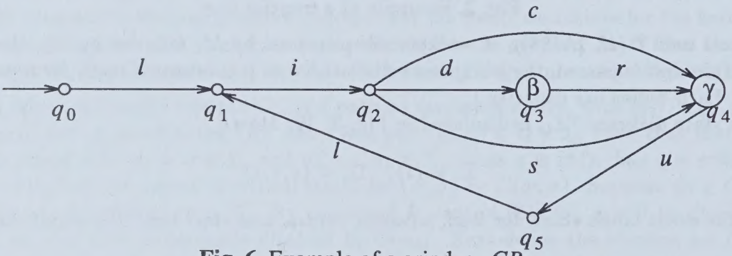


Fig. 6. Example of a grinder - GR_{lo}

generic for any simple industrial processor. The events are l (load a workpiece), i (start the process), c (complete the process), d (detect a flaw or an error), r (reject a workpiece), s (manual stop), u (unload), with

$$\Sigma_c = \{l, r, s\}, \quad \Sigma_u = \{i, d, c, u\}.$$

The normal work cycle is (l, i, c, u). The significant events are taken initially to be β (corresponding to d) and γ (to any of $\{c, r, s\}$). This low-level model is already OCC, with state-outputs β, γ each controllable. The high-level model GR_{hi} is shown in Fig. 7. It is unsatisfactory in that β and γ appear to be independently controllable, whereas the actual disablement of state-output β in GR_{lo} by command & control from either the initial state or from q_5 (following the occurrence of γ in GR_{hi}) would entail as an unexpected side effect the disablement of γ via operator's disablement in GR_{lo} of l: the agents (q_0, l) and (q_5, l) are inadmissible. We cure the situation by application of the algorithm sequence PHCC;POCC to GR_{lo}, with the result $GR_{lo,new}$ displayed, after state-set minimization, in Fig. 8. The state

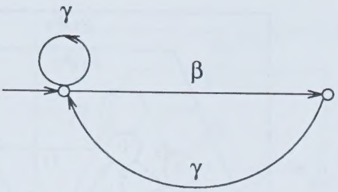


Fig. 7. GR_{hi}

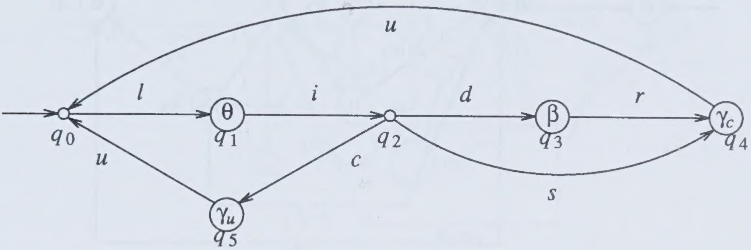


Fig. 8. $GR_{lo,new}$

q_1 was vocalized (with output θ, representing the significant event that loading is complete) and OCC then required that γ be split into controllable and uncontrollable instances γ_c, γ_u. The high-level result is $GR_{hi,new}$ as shown in Fig. 9 with β now uncontrollable. With the output θ, the manager is able to find the yield ratio of acceptable product of the grinder by counting the number of events γ_u and θ.

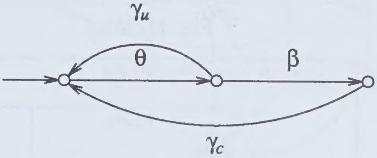


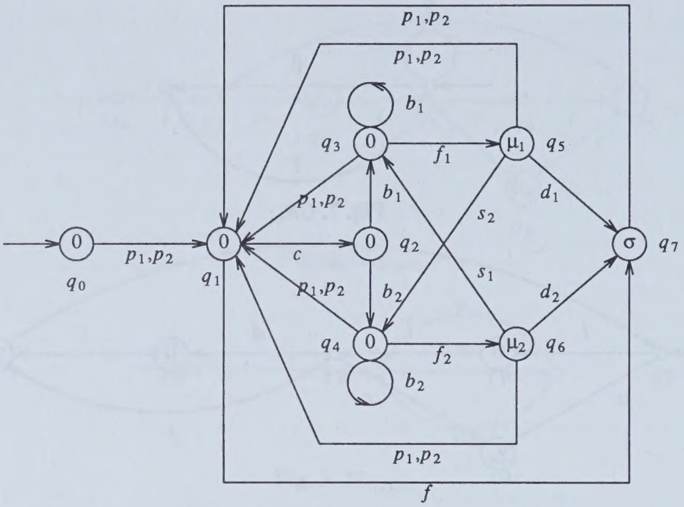
Fig. 9. $GR_{hi,new}$

3.3 Insertion Machine

The structure IM_{lo} (Fig. 10) represents a machine for inserting electronic components on printed circuit cards of two types. It is simplified from the description in [ACG84]. A card is selected via events (p_1, p_2), conveyed (c) and sorted to the appropriate buffer $(b_1$, $b_2)$; the machine enters process mode 1 or 2 via events f_1, f_2 and can switch modes via events s_1, s_2; additional events are breakdown $(d_1$, $d_2)$ and operator stoppage (f); the control status of these events is given by

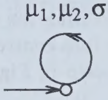
$$\Sigma_c = \{c, p_1, p_2, f\}, \quad \Sigma_u = \{b_1, b_2, f_1, f_2, s_1, s_2, d_1, d_2\}$$

Initially the significant high-level events, as suggested by the hierarchical production scheduling policy of [ACG84], are taken to be entrance to process mode 1 or 2 $(\mu_1$, $\mu_2)$ and breakdown (σ). We now address the issue of whether these events convey sufficient *managerial* information for hierarchical control on the basis of a managerial specification. As matters stand, the model IM_{hi} (Fig. 11) is unstructured; also inspection of IM_{lo} reveals that the control status of σ is ambiguous. This unpromising situation is rectified by application to IM_{lo} of the algorithm sequence POCC;PHCC;POCC. The initial result for POCC (Fig.12) has forced a



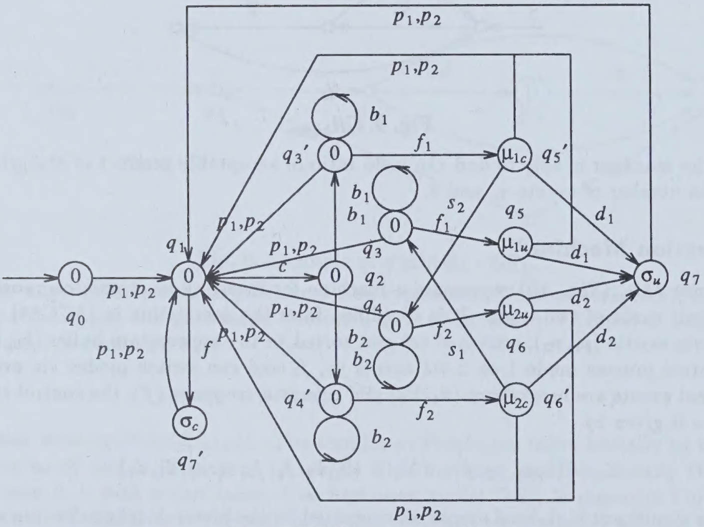
p_1,p_2
p_1,p_2
b_1
q_3
0
f_1
μ_1
q_5
p_1,p_2
b_1
d_1
s_2
0
p_1,p_2
0
c
0
q_2
σ
q_7
q_0
q_1
s_1
p_1,p_2
b_2
d_2
q_4
0
f_2
μ_2
q_6
b_2
p_1,p_2
f

Fig. 10. Example of an insertion machine - *IM_{lo}*



μ_1,μ_2,σ

Fig. 11. *IM_{hi}*



p_1,p_2
p_1,p_2
b_1
q_3'
0
f_1
μ_{1c}
q_5'
b_1
b_1
s_2
q_5
d_1
p_1,p_2
0
f_1
μ_{1u}
q_1
p_1,p_2
q_3
d_1
0
0
c
0
q_2
σ_u
q_7
p_1,p_2
q_0
p_1,p_2
q_4
d_2
μ_{2u}
p_1,p_2
0
f_2
b_2
s_1
q_6
d_2
p_1,p_2
f
b_2
q_4'
0
μ_{2c}
q_6'
f_2
σ_c
b_2
q_7'
p_1,p_2

Fig. 12. POCC result of Fig. 10

split of each of μ_1, μ_2, σ into controllable and uncontrollable instances: for instance σ_c signals operator stoppage, while σ_u signals machine breakdown. However, μ_{1c} and μ_{2c} are not independently controllable, as the agent (q_1, c) is inadmissible. In the final result for $IM_{lo,new}$ (Fig. 13) the state following c has been vocalized (with output θ) and the controllable instances of μ have been eliminated. The final managerial model $IM_{hi,new}$ (Fig. 14) can be

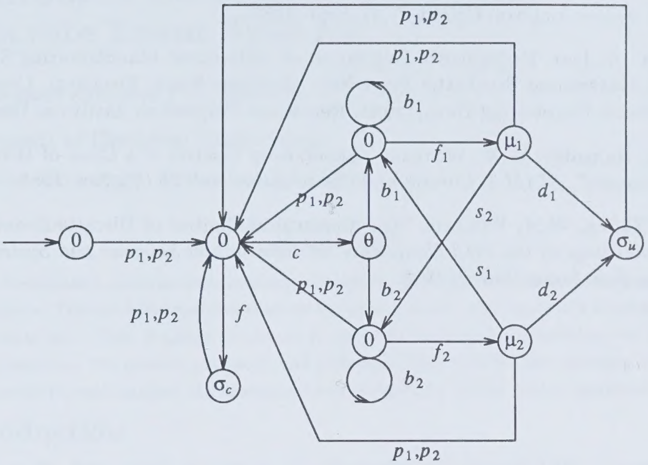


Fig. 13. $IM_{lo,new}$

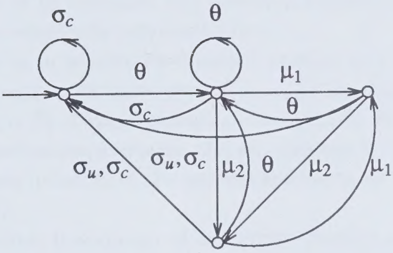


Fig. 14. $IM_{hi,new}$

employed consistently for command & control. For instance, a managerial specification that μ_1, μ_2 should alternate can be shown to be synthesizable in the hierarchical control loop.

4 Conclusion

As in [ZW88] we emphasize that achieving consistent hierarchical control requires careful attention to the selection of *significant* events and the analysis of their controllability status. Even for small examples like the Insertion Machine, the procedure to be followed could be quite difficult to identify by inspection; the same is true of the correctness of the result. The systematic algorithmic approach sketched in this paper and its predecessor provides a useful tool for the structural understanding of hierarchies from a formal control-theoretic point of view. The approach is capable both of refinement in the present framework (for instance by the incorporation of state marking) and of generalization to more complex and realistic logic models of discrete-event systems (such as timed transition models with stochastic attribute). These topics will be examined in future reports.

References

[ACG84] R. Akella. Y. Choong, S.B. Gershwin. "Performance of Hierarchical Production Scheduling Policy", *IEEE Transaction on Components, Hybrids, and Manufactur-*

ing Technology, vol. CHMT-7 (3), Sept. 1984.

[AJ88] R.Y. Al-Jaar. Performance Evaluation of Automated Manufacturing Systems Using Generalized Stochastic Petri Nets, *Seminar Notes*, Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Dec. 15, 1988.

[RW87] P.J. Ramadge, W.M. Wonham. "Supervisory Control of a Class of Discrete Event Processes", *SIAM I. Control and Optimization*, vol. 25 (1), Jan. 1987.

[ZW88] H. Zhong, W.M. Wonham. "On Hierarchical Control of Discrete-Event Systems", *Proceedings of the 1988 Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1988.

Asynchronous Coordination Problems for Discrete Event Systems

C.H. Golaszewski and P.J. Ramadge *

Department of Electrical Engineering

Princeton University, Princeton NJ 08544

Abstract

We formulate a general coordination problem for product discrete event systems modeled by automata. This problem requires that control maintain the invariance of a Boolean predicate on the state set. This Boolean predicate is specified in a modular fashion via a separable boolean function. We present necessary and sufficient conditions for the existence of a solution to this problem, and analyze the computational complexity of the control synthesis problem.

1. Introduction

Following the framework set out in [4], Ramadge [2] examined the computational complexity of several control synthesis problems for a special class of DES models called product systems. The main results of [2] indicated that certain interesting coordination problems for this class of systems where solvable in polynomial-time.

Our goal in this paper is to provide an extended analysis of a computationally tractable class of coordination problems under fairly general assumptions. Towards this end we present two extensions to the work in [2]. First, we formulate and provide conditions for the solution of a more general class of coordination problems. Second, contrary to the assumption made in [2] we allow the joint open-loop behavior of the product system to be "unfair" (this is explained in more detail in section 2).

Roughly a product system is a system of concurrent discrete event systems modelled by forming the shuffle product of the component subsystems. Since the number of states in the product system increases exponentially with the number of component processes, the key issue is computational complexity.

A coordination problem for a product system is a control problem that arises when the joint operation of the processes is subject to global constraints. In such situations not all possible interleavings of the event trajectories generated by the component processes are allowed. There are two problems to be addressed here. First, it is necessary to specify the control constraint in a tractable fashion (in polynomial space). The objective is then to synthesize (in polynomial-time) a controller for the product system which ensures this constraint is met.

The results in this paper are based on [1]. We assume familiarity with the recent work on so-called logical models for discrete event systems. Readers not having prior exposure to this material might find the survey paper [3] useful prerequisite reading.

2. Product Systems

We consider a discrete event system that consists of p components operating in parallel, with each component described by a controlled finite state generator

$$G_i = (\Sigma_i, Q_i, \delta_i, q_{oi}, Q_{im}),$$

---

* Research partially supported by the National Science Foundation through grant ECS87-15217 and by an IBM Faculty Development Award.

with control partition $\Sigma_i = \Sigma_{iu} \cup \Sigma_{ic}$, $i \in [1,p]$. We assume that the event sets Σ_i are *non-redundant*, i.e., for all $\sigma \in \Sigma_i$ there exists a state $q \in Q_i$ such that $\delta_i(\sigma, q)!$, and that for $i \neq j$, $\Sigma_i \cap \Sigma_j = \emptyset$. For each G_i, Q_i is the state set, $\delta_i : \Sigma_i \times Q_i \to Q_i$ is the state transition function, q_{oi} is the initial state, and $Q_{im} \subseteq Q_i$ is a set of marker states.

Let $\Sigma_i^{\mathbb{N}}$ denote the set of of infinite sequences of elements of Σ_i. We display $e \in \Sigma_i^{\mathbb{N}}$ as $e = e(1)e(2)e(3)\ldots$, and let e^j denote the truncation of e to the first j events, i.e., $e^j = e(1)e(2)\ldots e(j)$.

The string behavior $L(G_i) \subseteq \Sigma_i^*$ of G_i is the set of finite length strings of events such that $\delta_i(s, q_{oi})$ is defined. The sequential behavior of G_i is the set

$$B(G_i) = \{e \in \Sigma_i^{\mathbb{N}} : \delta(e(j), q_{oi}) \in Q_{im} \text{ for infinitely many } j \in \mathbb{N}\}$$

The concurrent operation of the G_i is modeled by the *synchronous product* $G = \|_{i=1}^{p} G_i$. This is the generator $G = (\Sigma, Q, \delta, q_o)$ with

$$\Sigma = \bigcup_{i=1}^{p} \Sigma_i \ , \quad Q = Q_1 \times \ldots \times Q_p \ , \quad q_o = (q_o(1), \ldots, q_o(p))$$

and state transition function $\delta : \Sigma \times Q \to Q$ defined by

$$\delta(\sigma, q) = \begin{cases} q' \ , \qquad q'(i) = \begin{cases} q(i), & \text{if } \sigma \notin \Sigma_i; \\ \delta(\sigma, q(i)), & \text{if } \sigma \in \Sigma_i \text{ and } \delta_i(\sigma, q(i))!. \end{cases} \\ \\ \text{undefined}, \quad \text{otherwise.} \end{cases}$$

The natural projection $\pi_i : \Sigma^* \to \Sigma_i^*$ is the map

$$\pi_i(\sigma) = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_i; \\ 1, & \text{if } \sigma = 1 \text{ or } \sigma \in \Sigma \text{ and } \sigma \notin \Sigma_i. \end{cases}$$

with $\pi_i(w\sigma) = \pi_i(w)\pi_i(\sigma)$ for $\omega \in \Sigma^*$ and $\sigma \in \Sigma$.

The behavior generated by the product system G is

$$L(G) = \{w \in \Sigma^* : \forall i \in [1,p] : \pi_i(w) \in L(G_i)\}.$$

The set of all infinite sequences of events having the property that every finite truncation of the sequence is an element of $L(G)$ is denoted by $L(G)^\infty$. A sequence $e \in \Sigma^{\mathbb{N}}$ is Σ_i-recurrent if $e(j) \in \Sigma_i$ for infinitely many indices $j \in \mathbb{N}$. In this case we denote the unique subsequence of e consisting of elements of Σ_i by e_i, and define $\pi_i : \Sigma^{\mathbb{N}} \to \Sigma_i^{\mathbb{N}}$ by

$$\pi_i(e) = \begin{cases} e_i, & \text{if } e \text{ is } \Sigma_i\text{-recurrent;} \\ \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

A sequential behavior $C \subseteq \Sigma^{\mathbb{N}}$ *fair* if for all $e \in C$ and all $i \in [1,p]$, $\pi_i(e)$ is defined. For example,

$$B'(G) = \{e \in L(G)^\infty : \forall i \in [1,p] : e \text{ is } \Sigma_i - \text{recurrent and } \pi_i(e) \in B(G_i)\}$$

is a fair sequential behavior (see [2]).

In contrast to [2] in this paper we take the *sequential behavior* $B(G)$ of the product generator G to be

$$B(G) = \{e \in L(G)^\infty : \forall i \in [1,p] : \pi_i(e) \in B(G_i) \text{ or } \pi_i(e) \text{ is undefined } \}.$$

This behavior is not fair. We make fairness part of the supervisory control synthesis problem.

For $e_i \in B(G_i)$, $i \in [1,p]$, we let $||_{i=1}^p\, e_i$ be the subset of $\Sigma^{\mathbb{N}}$

$$||_{i=1}^p\, e_i = \{e \in L(G)^\infty : \pi_i(e) = e_i, i \in [1,p]\}.$$

We partition Σ into the sets $\Sigma_u = \cup_{i=1}^p \Sigma_{ui}$ and $\Sigma_c = \cup_{i=1}^p \Sigma_{ci}$. As usual we let

$$\Gamma = \{\gamma : \Sigma_u \subseteq \gamma\}$$

denote the set of control inputs for G. A supervisor for G is a map $f: L(G) \to \Gamma$ that specifies the input to be applied to the product system after the current event string is observed. For a detailed interpretation of this form of control see [3]. If a supervisor f is applied to G, then the resultant closed loop behaviors are denoted by $L(G,f)$ and $B(G,f)$. These are defined by:

(1) $1 \in L(G,f)$;

(2) $s\tau \in L(G,f)$ if and only if $s \in L(G,f)$, $\tau \in f(s)$ and $s\tau \in L(G)$; and

(3) $B(G,f) = L(G,f)^\infty \cap B(G)$.

We say that a supervisor f is nonblocking for G if

$$pr(B(G,f)) = L(G,f).$$

This means that every string $s \in L(G,f)$ is a prefix of a sequence in $B(G,f)$ and hence the closed-loop system is 'dead-lock' free. We assume that the individual generators G_i are nonblocking.

3. Coordinators

Attention is restricted to the subbehaviors of $B(G)$ which we call coordinated behaviors. Roughly these behaviors result when G is controlled by a supervisor that is only permitted to delay events relative to the occurrence of others. Formally, $C \subseteq B(G)$ is a *coordinated behavior* if C is fair and

$$\forall e_i \in B(G_i), i \in [1,p] : ||_{i=1}^p\, e_i \cap C \neq \emptyset.$$

The supervisors for G that result in a coordinated closed loop behavior are called *coordinators.* We say that a coordinator f' is *minimally restrictive* for a behavior C if $B(G,f) \subseteq B(G,f')$ for all coordinators f satisfying $B(G,f) \subseteq C$. The following proposition is useful for demonstrating the existence of a minimally restrictive coordinator. A proof can be found in [1]. Similar results are presented in [2].

Proposition 3.1. *Let G be a product system with behavior $B(G)$, and let $C \subseteq B(G)$. If C is topologically closed with respect to $B(G)$, i.e., $pr(C)^\infty \cap B(G) = C$, and there exists a coordinator such that $B(G,f) \subseteq C$, then there exists a maximal coordinator f with $B(G,f) \subseteq C$.*

4. Coordination Problems

In our context a supervisory control problem for the product system G could be formulated as follows. For a given behavior $C \subseteq B(G)$, find a nonblocking coordinator f such that $B(G,f) \subseteq C$. Since we are particularly interested in the computational complexity of solving such synthesis problems we must pay attention to how the elements of the problem are specified. The product generator is given in a modular fashion. This allows us to specify in polynomial space (polynomial in p and n) a complex system with an exponential number of states (exponential in p). However, as things stand we have no concise means of specifying C.

One way to define C is to bring in a partition of the state space Q of the product system G into nonempty disjoint sets Q^- and Q^+. The set Q^+ is the "good set"; all state trajectories that completely remain in Q^+ correspond to permitted event sequences. This partition of the state space of G can be specified by means of Boolean functions. Let Q_i be the state set of G_i and define a map $h_i : Q_i \to \mathbb{B}$, where $\mathbb{B} = \{0,1\}$. Denote the set of states $q \in Q_i$ such that $h_i(q) = 1$ by Q_i^+ and let $Q_i^- = Q_i - Q_i^+$. We say that a subset $M_i \subseteq Q_i$ is *nontransient* if there exists a sequence $e_i \in B(G_i)$ such that the corresponding state trajectory of G_i enters the set M_i infinitely often. We assume that the sets Q_i^+, Q_i^- are nontransient.

The maps h_i associate an element of $\mathbb{B}^p$ with every state of the product system G. For $q \in Q$ define

$$h_i(q) = \begin{cases} 1, & \text{if } q(i) \in Q_i^+; \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

To define the division of the state space we could let $h : \mathbb{B}^p \to \mathbb{B}$ be a Boolean function, and set $Q^+ = \{q \in Q : h(h_1(q), \ldots, h_p(q)) = 1\}$. However, we then need to discuss how the function h is specified. Simply listing its truth table would require 2^p entries.

Let $\mathbf{Q}$ denote the set of rational numbers. The domain of the function h, i.e., the vectors in $\mathbb{B}^p$, can be viewed as the corners of a hypercube in $\mathbf{Q}^p$. A special case arises if the function h is separable by hyperplanes. A hyperplane H in $\mathbf{Q}^p$ is parametarized in the form

$$\sum_{i=1}^{p} \lambda_i x_i = r, \quad \lambda_i, r \in \mathbf{Q}, i \in [1,p]. \tag{$*$}$$

Without loss of generality we may assume that $\lambda_i \neq 0$, $i \in [1,p]$.

We say that the Boolean function h is m-separable if there exist m hyperplanes

$$H_j : \sum_{i=1}^{p} \lambda_{ij} x_i = r_j, \qquad j \in [1,m]$$

such that

$$h^{-1}(1) = L^+ = \{b \in \mathbb{B}^p : \sum_{i=1}^{p} \lambda_{ij} b_i \leq r_j, j \in [1,m]\}.$$

Under suitable assumptions on the number of bits used to express the rational coefficients for each hyperplane we can ensure that an m-separable boolean map h is specified in polynomial-space.

We now pose the Boolean coordination problem for p processes and an m-separable Boolean function:

BCP(p,m): Let $h_i : Q_i \to \mathbb{B}$ be defined as above, and assume that for each i, $Q_i^+ \subseteq Q_i$ is a nontransient subset. Given an m-separable Boolean function $h : \mathbb{B}^p \to \mathbb{B}$, find a nonblocking coordinator such that for all $e \in B(G,f)$ the corresponding state trajectory of G remains in $Q^+ = \{q \in Q : h(h_1(q), \ldots, h_p(q)) = 1\}$.

The conditions $\sum_{i=1}^{p} \lambda_{ij} b_i \leq r_j$ that are required to be satisfied in **BCP(p,m)** are called the *coordination constraints.*

5. The Simplest Case

We first investigate the special case BCP(p,1). In this case the function h is separable by a single hyperplane. Define the sets $L^+ \subseteq \mathbb{B}^p$ and $L^- \subseteq \mathbb{B}^p$ by $L^+ = \{b \in \mathbb{B}^p : h(b) = 1\}$, and $L^- = \mathbb{B}^p - L^+$. Geometrically, the hyperplane divides the hypercube in $\mathbb{R}^p$ with corners in $\mathbb{B}^p$ into two parts which correspond to the two convex sets L^+ and L^-.

5.1 Critical Sets

Let $I^+ = \{i : \lambda_i > 0\}$ and $I^- = \{i : \lambda_i < 0\}$. Then for $i \in I^+$ set

$$D_i = \{q \in Q_i : \delta_i(w,q) \in Q_i^+ \text{ for some } w \in \Sigma_{ui}^*\};$$

and for $i \in I^-$ set

$$D_i = \{q \in Q_i : \delta_i(w,q) \in Q_i^- \text{ for some } w \in \Sigma_{ui}^*\}.$$

Once a process G_i is inside its set D_i it can reach the set Q_i^+ (or Q_i^- if $i \in I^-$) via strings of uncontrolled events. On the other hand, if the process is in the set $Q_i - D_i$ then it is possible to control G_i so that the state trajectory does not enter D_i, i.e., $Q_i - D_i$ is controlled-invariant. The following maximal supervisor for the product system prevents G_i from entering D_i:

$$f_i(e^j) = \bigcup_{i=1}^{p} \Sigma_i - \{\alpha \in \Sigma_{ic} : \delta_i(\pi_i(e^j), q_o(i)) \notin D_i \text{ and } \delta_i(\pi_i(e^j)\alpha, q_o(i)) \in D_i\}.$$

For future reference we note ([2]):

Lemma 5.1. *The sets D_i can be determined in a time bounded by a polynomial in p and $\max_i |Q_i|$.*

The sets D_i play a crucial role in the solution of BCP. To see this define functions $d_i : Q \to \mathbb{B}$ by

$$d_i(q) = \begin{cases} 1, & \text{if } q(i) \in D_i \text{ and } i \in I^+; \\ 1, & \text{if } q(i) \in Q_i - D_i \text{ and } i \in I^-; \\ 0, & \text{otherwise.} \end{cases}$$

We then have:

Proposition 5.1. *An instance of the Boolean coordination problem is solvable if and only if it is solvable with the functions d_i replacing the functions h_i.*

Proof. (if). Suppose there exists a coordinator f which solves the Boolean coordination problem with the d_i's replacing the h_i's. Since $\lambda_i h_i \leq \lambda_i d_i$, $\sum_{i=1}^{p} \lambda_i d_i(q) \leq r \Rightarrow \sum_{i=1}^{p} \lambda_i h_i(q) \leq r$. Thus the coordinator f also solves the original instance.

(only if). Suppose the coordinator f solves BCP(p,1), i.e., f guarantees that the closed loop system satisfies $\sum_{i=1}^{p} \lambda_i h_i(q) \leq r$. Assume there exists a sequence $e \in B(G,f)$ such that for some j, $\sum_{i=1}^{p} d_i(\delta_i(e^j, q_o)) > r$. Since f is a solution to BCP we must have $\sum_{i=1}^{p} h_i(\delta_i(e^j, q_o)) \leq r$. But by the definition of D_i and d_i there exists an uncontrollable string $s \in \Sigma^*$ such that after $e^j s$, $\lambda_i d_i$ has the same value as after e^j, but after $e^j s$ we have $\lambda_i h_i = \lambda_i d_i$ for each i. Thus $\sum_{i=1}^{p} h_i(\delta_i(e^j s, q_o)) > r$; is a contradiction. Hence f also solves BCP with the d_i replacing h_i. ◇

0.1 Normal Form

For the remainder of this section we assume without loss of generality that $Q_i^+ = D_i$ if $i \in I^+$, $Q_i^- = D_i$ if $i \in I^-$, and hence that $d_i = h_i$, $i \in [1,p]$.

We say that an instance of BCP(p,1) is in *normal form* if $I^- = \emptyset$. For a given instance of BCP(p,1) we can construct an equivalent instance in normal form as follows: Define functions $g_i : Q \to \mathbb{B}$ according to

$$g_i(q) = \begin{cases} 1, & \text{if } q(i) \in D_i\text{; and} \\ 0, & \text{otherwise.} \end{cases}$$

Obviously, if $i \in I^-$, then g_i is the complement of h_i. The functions g_i allow the new but equivalent parametarization of the constraint region:

$$\sum_{i=1}^{p} |\lambda_i| g_i(q) \leq r + \sum_{j \in I^-} |\lambda_j|$$

In this parametarization all the coefficients of the hyperplane are positive. Clearly a solution to an instance of BCP(p,1) can be found by solving the equivalent instance of BCP(p,1) in normal form.

0.2 Solution to BCP(p,1)

Our solution to BCP(p,1) has two components. First, the subsupervisors f_i that keep $Q_i - D_i$ invariant for each of the individual generators are used to ensure that the coordination constraints are met. To ensure fairness, the application of these component supervisors is regulated using certain information gleaned from the event trajectories concerning possible future entry into the sets D_i (we call these 'request states') together with a queue. The idea is that if a process enters its set of 'request states', it is placed in the queue to await entry into its D_i.

As usual, the Hamming distance $\mu(b, b')$ of two vectors $b, b' \in \mathbb{B}^p$ is the number of indices i such that $b(i) \neq b'(i)$.

Our first result is the following proposition. It extends Proposition 5.1 in [2] to BCP(p,1):

Proposition 0.1. *Consider an instance of BCP(p,1) in normal form. Suppose that for all* $i \in [1,p]$, *the generator* G_i *has no cycles in* D_i *that intersect* Q_{im}. *Then there exists a nonblocking coordinator that solves this instance of the Boolean coordination problem if and only if*
1.) the initial state q_o *is in* Q^+*; and*
2.) $\{b \in \mathbb{B}^p : \mu(b, (0, \ldots, 0)) = 1\} \subseteq L^+$.

We display below a supervisor that solves BCP when the above conditions are met. As stated above the supervisor has two components. For $i \in [1,p]$ define the *set of request states* of G_i to be the subset of Q_i given by

$$R_i = \{q \in Q_i - D_i : \exists \sigma \in \Sigma_i \text{ such that } \delta_i(\sigma, q) \in D_i\}.$$

When a subsystem G_i enters its set of request states its index i is placed in an first come ordered list. This can be thought of as a queue. When G_i enters D_i, or when G_i leaves its set of request states, the index i is removed from the queue. Let $\eta \in \mathbb{N}$ denote the index of the first process in the queue. We assume that $\eta = 0$ if the queue is empty.

Define the supervisor f by

$$f(e^j) = \begin{cases} \Sigma, & \text{if } \eta = 0; \\ \bigwedge_{\{i \neq \eta : b_i = 0\}} f_i(e^j), & \text{if } \sum_{i=1}^{p} \lambda_i b_i + \lambda_\eta \leq k. \\ \bigwedge_{\{i : b_i = 0\}} f_i(e^j), & \text{if } \sum_{i=1}^{p} \lambda_i b_i + \lambda_\eta > k. \end{cases} \qquad (*)$$

The idea behind the supervisor is this. Entry into the sets D_i is regulated so that the coordination constraint is met. Since $Q_i - D_i$ is controlled-invariant this can be achieved by the component supervisors f_i. This is the same basic strategy used in [2]. However, to ensure fairness of the closed loop system we need to keep track of potential entry in the sets D_i and ensure that each process with the potential to enter D_i is eventually given the chance to do so. This is the role of the request states and the ordered list. Each request state is a state from which the process can enter its D_i, the queue keeps track of the order in which these states are reached. This order is used to select which process is to be allowed entry into its D_i when it is safe to do so. The index of a process is removed from the queue if it leaves its set of request states (in which cases it forfeits its priority for entry), or when entry is granted and taken.

It is intuitively clear that the supervisor will ensure theat the coordination constraint is met. The only issue that needs to be resolved is whether or not the supervisor is a nonblocking coordinator. This is where the assumptions of the propostion are used. The verification is tedious but straightforward. Full details can be found in [1]. A similar argument for a simpler case can also be found in [2].

The following theorem contains our main result on BCP(p,1):

Theorem 5.1. *Consider an instance of BCP(p,1). It can be verified in polynomial-time if this instance is solvable. Furthermore, for each solvable instance a nonblocking coordinator can be constructed in a time bounded by a polynomial in p and* $\max_i Q_i$.

Proof. The sets D_i and the functions d_i can be constructed in polynomial-time. Clearly, any linearly separable instance can be transformed into normal form in polynomial-time. It is straightforward to check if there exists an i such that D_i contains a cycle intersecting Q_{im} (see [2]). The necessary and sufficient conditions for solvability can also be checked in a time bounded by a polynomial in p and $\max_i Q_i$.

To construct the supervisor f defined above one has to construct the supervisors f_i. These can be found in polynomial-time from the sets Q_i and D_i. The request states can be easily found from the set D_i and hence the queue can be implemented in polynomial-time. The implementation of the sum is straightforward. ◊

6. Conclusion

The Boolean coordination problem was introduced as a means of concisely specifying certain contraints on the concurrent operation of the components of a product discrete event system. The mutual exclusion problem analyzed by Ramadge in [2] is a special case of this problem. It was shown that instances of BCP(p,1) can be solved in a time bounded by a polynomial in the input size. A discussion of the more general problem BCP(p,m), can be found in [1]. There it is shown that the decidability of the solvability of the general problem is NP-complete. However, several special cases are shown to be solvable in polynomial time.

References

[1] Golaszewski, C.H., "On the Supervisory Control of Discrete Event Systems", *Ph.D. Thesis*, Princeton University, September 1989.

[2] Ramadge, P.J., "Some Tractable Supervisory Control Problems for Discrete Event Systems Modeled by Büchi Automata", *IEEE Transactions on Automatic Control*, Vol. 34 (1), January 1989.

[3] Ramadge, P.J., and Wonham, W.M., "The Control of Discrete-Event Systems", *IEEE Proceedings*, Vol.77(1), January 1989.

[4] Ramadge, P.J., and Wonham, W.M., "Supervisory Control of a Class of Discrete-Event Processes", *SIAM J. Control and Optimization*, Vol. 25(1), January 1987.

LINEAR INTEGER PROGRAMMING TECHNIQUES IN THE CONTROL OF VECTOR DISCRETE-EVENT SYSTEMS

Y. LI and W.M. WONHAM

Systems Control Group
Department of Electrical Engineering
University of Toronto
Toronto, CANADA M5S 1A4

ABSTRACT

A vector discrete-event system (VDES) is modelled on an n-dimensional vector state space generated by n state variables, with a state transition function determined by displacement vectors and occurrence vectors. The characterization of the reachable state set of a VDES is first discussed. It is shown that this reachable set can be characterized by feasible solutions to a system of n linear inequalities, if the VDES under consideration has loop-free structure. Using this result we transform the problem of computing the optimal control function into a standard linear integer programming problem, which can be solved efficiently using existing optimization techniques. The method is extended to the synthesis of controllers with partial state information.

1. INTRODUCTION

Discrete-event systems (DES) are those which are discrete, asynchronous and possibly nondeterministic. Many technological systems provide examples -- e.g. computer networks, flexible manufacturing systems, and traffic control systems.

The study of DES has been extensive in the last decade or so, but mainly concentrated in the computer community [1,2]. The investigation of DES from a control-theoretic viewpoint was initiated by Ramadge and Wonham [11,14], where a DES was modeled as a state-transition system (an automaton), while the controller, or supervisor, for this DES enforces and is constructed from a legal-behavior specification (legal language) of the system. This constructive approach is characteristic of control theories and contrasts with the approach more common in computing theory, which tends to emphasize evaluation and verification of tentative solutions.

However, one problem encountered in the constructive approach is computational efficiency. For large DES this problem becomes more prominent. A flexible manufacturing system may have such a large state space that a construction based on processing of a listed transition structure is infeasible. Fortunately, many DESs in actuality have structured state spaces and structured transition functions. By exploiting the structure information we may be able to find solutions to various control problems which can be computed more efficiently -- regardless of system size.

Motivated by this observation, a class of structured DES, called *vector DES* (VDES), was studied in [4] The state space of a VDES is an n-dimensional integer vector space generated by n state variables, and the state transition function is determined by *displacement vectors* and *occurrence vectors* in the state space.* The state feedback control of VDES was studied in [4], with emphasis on linear legal-state specifications. It was shown that if the uncontrollable part of VDES has a "tree-like" structure then the optimal state-feedback function can be efficiently computed without resorting to state-graph search. However, the tree-likeness requirement on a VDES is too restrictive for applications.

A more general structure we study in this paper is loop-free (LF) structure. We first show that if a VDES is LF then the set of states reachable from any given state can be characterized by feasible solutions to a system of n linear inequalities (Section 3). Using this result we prove in Section 4 that if the legal state specification is given in the form of a linear predicate then the problem of computing the optimal control function can be transformed into a standard linear integer programming problem, which can be solved using existing optimization techniques. We will also discuss how to reduce the above integer programming problem to one of lower dimension. The reduction allows us to obtain an observability subspace. Only partial state information corresponding to this observability subspace is needed in computing the optimal control function. An illustration is provided in Section 5.

---

* Strictly speaking, the state space is a "module over the ring of integers" and not a vector space; but we adopt vector space terminology anyhow.

2. CONTROL OF VECTOR DISCRETE-EVENT SYSTEMS

The state of a *vector discrete-event system* (VDES) is described by *state variables* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, ranging over integers $\mathbf{Z}$. The state space of a VDES is defined as the set of all n-dimensional integer vectors:

$$\mathbf{X} := \mathbf{Z}^n = \{X=[x_1,x_2,\ldots,x_n] \mid x_i \in \mathbf{Z}\}.$$

For each event α in the *event set* Σ the *occurrence condition* $C(\alpha,.)$: $\mathbf{X} \rightarrow \{0,1\}$ is a predicate on $\mathbf{X}$ specified by

$$C(\alpha,X)=1 \Leftrightarrow X \geq F_\alpha$$

where $F_\alpha \in \mathbf{X}$. The state transition function $\delta: \Sigma \times \mathbf{X} \rightarrow \mathbf{X}$ is given by

$$\delta(\alpha,X)=X+E_\alpha$$

with $E_\alpha \in \mathbf{X}$. We impose only a mild condition on the transition function:

$$(\forall X \in \mathbf{X})\ X \geq F_\alpha \Rightarrow X+E_\alpha \geq 0$$

which guarantees that the system has nonnegative state components when starting with nonnegative initial state components. Without loss of generality, we assume in the sequel that $F_\alpha = -E_\alpha$ for $\alpha \in \Sigma$. In other words we assume that α can occur at X if and only if the resultant state after α occurs has nonnegative state components. To further simplify discussion we also assume that E_α has components 0,1, or -1. The results obtained in this paper can be extended easily to the general case. Finally, our VDES is described by the following 5-tuple:

$$\mathbf{G}=(\Sigma, \mathbf{X}, \delta, C, X_0)$$

where X_0 is the initial state. Both δ and C can be extended to be defined on $\Sigma^* \times \mathbf{X}$: $C(w,X)=1$ means that the event string w can occur from X; and after the occurrence of w the resultant state is $\delta(w,X)$.

Definition 2.1

The *feasible language* $L(\mathbf{G},X)$ for a state X in $\mathbf{G}$ is defined as

$$L(\mathbf{G},X) := \{w \in \Sigma^* \mid C(w,X)=1\}.$$

The *reachable-state set* $R(\mathbf{G},X)$ from a state X in $\mathbf{G}$ is defined as

$$R(\mathbf{G},X) := \{X'=\delta(w,X) \mid w \in L(\mathbf{G},X)\}.$$

A control mechanism is incorporated into $\mathbf{G}$ by partitioning Σ into two disjoint subsets:

$$\Sigma = \Sigma_c \cup \Sigma_u;$$

Σ_c and Σ_u are called the *controllable* and *uncontrollable event* sets respectively [11]. A controller can enable or disable a controllable event, while it cannot influence an uncontrollable event. The set of control commands, called *control patterns*, is defined as

$$\Gamma = \{\gamma \in 2^\Sigma \mid \gamma \supseteq \Sigma_u\}.$$

A control pattern γ specifies the events enabled by a controller at a given time. A *state feedback controller* is a function f: $\mathbf{X} \rightarrow \Gamma$.

For a VDES $\mathbf{G}$ and a predicate P with $X_0 \in P$, define a set of states reachable from X_0 inductively as:

(1) $X_0 \in Re(\mathbf{G},P)$;

(2) If $X \in Re(\mathbf{G},P)$, $\delta(\alpha,X)!$ for some $\alpha \in \Sigma$ and $\delta_\alpha(X) \in P$ then $\delta(\alpha,X) \in Re(\mathbf{G},P)$;

(3) Every $X \in Re(\mathbf{G},P)$ is obtained as in (1) or (2).

In particular, $Re(f/\mathbf{G},\mathbf{X})$ includes all and only states reachable from X_0 in the closed-loop system $f/\mathbf{G}$. Call $R(f/\mathbf{G}) := Re(f/\mathbf{G},P)$ the *closed-loop predicate*. It can be proved [5] that there exists a controller f such that

$$R(f/\mathbf{G})=P$$

if and only if P is *controllable*. P is controllable if

$$(\forall \alpha \in \Sigma_u)\ P \ \leq \ Re(\mathbf{G},P) \wedge wlp_{\alpha}(P).$$

It can also be shown that the supremal element $supC(P)$ in the set

$$\{ P' \ \mid \ P' \leq P,\ P' \text{ is controllable}\}$$

exists. Define

$$[P] := \{X \ \mid \ (\forall w \in \Sigma_u^*)\ \delta(w,X)! \Rightarrow \delta(w,X) \in P\}.$$

Then we have [4]

$$supC(P) = Re(\mathbf{G},[P]),$$

and

$$R(f^* / \mathbf{G}) = supC(P)$$

with

$$f^*(\alpha,X) := \begin{cases} 1 & \text{if } \delta(\alpha,X) \in [P] \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the control function of the optimal controller f^* is determined by $[P]$.

So far we have been assuming that all state variables can be measured and made available to a controller. We now discuss state feedback control with partial state information. Without loss of generality, assume that the controller can only observe state variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n'}$, $n' \leq n$. We call

$$\mathbf{Y} := \{Y = [x_1, x_2, \ldots, x_{n'}] \ \mid \ x_i \in \mathbf{Z}\}$$

the *observability subspace* of the system. Then

$$\mathbf{X} = \mathbf{Y} \oplus \overline{\mathbf{Y}}$$

for a subspace $\overline{\mathbf{Y}}$ of dimension $n-m$. Let $T: \mathbf{X} \rightarrow \mathbf{Y}$ be the natural projection from $\mathbf{X}$ to $\mathbf{Y}$ and T^{-1} its inverse image function.* Controllers with partial state information are defined as

$$\mathbf{F}_o := \{f \ \mid \ (\forall X,X')(\forall \alpha \in \Sigma)\ T(X) = T(X') \ \Rightarrow \ f(\alpha,X) = f(\alpha,X')\}. \tag{2.1}$$

It was proved [5] that there exists a controller $f \in \mathbf{F}_o$ such that

$$R(f/\mathbf{G}) = P$$

if and only if P is controllable and *observable* wrt $\mathbf{Y}$. P is observable wrt to $\mathbf{Y}$ if

$$(\forall \alpha \in \Sigma_c)\ P \geq De(\alpha,P) \wedge sp_{\alpha}(P)$$

where

$$De(\alpha,P) := T^{-1}(T(sp_{\alpha}(P) \wedge P).$$

In the sections to follow we discuss the efficient controller synthesis by exploiting system structure.

3. CHARACTERIZATION OF REACHABLE STATES IN VDES

In this section we show that when a VDES $\mathbf{G}$ has a loop-free structure the reachable state set $R(\mathbf{G},X)$ can be characterized by feasible solutions to a system of linear inequalities. This result will be used in the next section for efficient controller synthesis.

For an event string $w \in \Sigma^*$ the *occurrence vector* $V_w \in \mathbf{Z}^m$ is defined as

$$V_w(i) = |w|_{\alpha_i}, \quad \alpha_i \in \Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_m{}'\}$$

where $|w|_{\alpha_i}$ is the number of α_i-occurrences in w. The occurrence vector V_w is sufficient to determine the

---

* It is easy to check that T thus defined satisfies the condition:

$$(\forall X,X')(\forall \alpha)\ T(X) = T(X') \Leftrightarrow T(\delta_{\alpha}(X)) = T(\delta_{\alpha}(X')).$$

This condition was used to obtain the results on observability in [5].

resultant state after the occurrence of w: $\delta(w,X) = X + E\,V_w$, with the $n \times m$ matrix E being defined as

$$E := [E_{\alpha_1}, E_{\alpha_2}, \ldots, E_{\alpha_{m'}}].$$

This implies that for any $w \in \Sigma^*$

$$w \in L(\mathbf{G},X) \Rightarrow X + E\,V_w \geq 0. \tag{3.1}$$

More significant is the reverse of (3.1), i.e.

$$X + E\,V \geq 0 \Rightarrow (\exists w \in L(\mathbf{G},X))\, V = V_w, \tag{3.2}$$

because if (3.2) holds then $R(\mathbf{G},X)$ can be nicely characterized:

$$\{ V_w \mid w \in L(\mathbf{G},X) \} = \{ V \mid X + EV \geq 0,\ V \geq 0 \}, \text{ and}$$

$$R(\mathbf{G},X) = \{X + E\,V \mid V \geq 0,\ X + E\,V \geq 0\}$$

It turns out that (3.2) does not always hold. Consider the following simple VDES **G**:

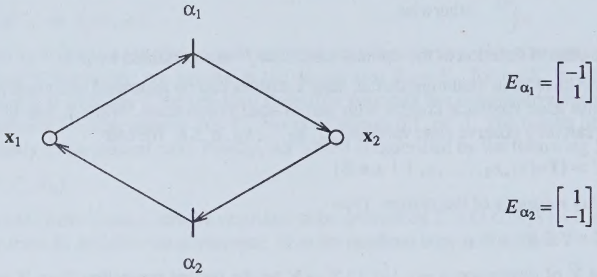

α_1
$E_{\alpha_1} = \begin{bmatrix}-1\\1\end{bmatrix}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$E_{\alpha_2} = \begin{bmatrix}1\\-1\end{bmatrix}$
α_2

Figure 3.1

Let $X = \begin{bmatrix}0\\0\end{bmatrix}$. Then

$$X + E\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}-1 & 1\\1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}.$$

But $V_{\alpha_1\alpha_2} = V_{\alpha_2,\alpha_1} = \begin{bmatrix}1\\1\end{bmatrix}$ and $\alpha_1\alpha_2, \alpha_2\alpha_1 \notin L(\mathbf{G},X)$.

In fact our main result of this section says that (3.2) holds if **G** is *loop-free* wrt Σ.

Definition 3.1

G is loop-free (LP) wrt $\Sigma' \subseteq \Sigma$ if we cannot find $\alpha'_0, \alpha'_1, \ldots, \alpha'_{t-1} \in \Sigma'$ and state variables $\mathbf{x}_{i_0}, \mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_{t-1}}$ such that

$$E_{\alpha'_k}(i_k) < 0 \text{ and } E_{\alpha'_k}(i_j) > 0,\ \ j := k+1 \ (\mathrm{mod}\, t) \quad (k = 0, 1, \ldots, t-1).$$

We now have:

Theorem 3.1

Assume that **G** is LF wrt Σ. Then for any state $X \geq 0$, $V \geq 0$,

$$X + EV \geq 0 \Leftrightarrow (\exists w \in L(\mathbf{G},X))\, V = V_w.$$

□

In the next section we will use Theorem 3.1 to transform the optimal controller synthesis problem into a linear integer programming problem.

4. COMPUTAION OF THE OPTIMAL CONTROL FUNCTION

As mentioned in Section 2 the supremal controllable subpredicate of a given predicate P is

$$supC(P) = \mathrm{Re}(\mathrm{G}, [P])$$

where

$$[P] := \{X \mid (\forall w \in \Sigma_u^*)\ \delta(w,X)! \Rightarrow \delta(w,X) \in P\}.$$

Furthermore, the optimal controller f^* synthesizing $supC(P)$ (i.e. $R(f^*/\mathrm{G}) = supC(P)$) can be computed by

$$f^*(\alpha,X) = \begin{cases} 1 & \text{if } \delta(\alpha,X) \in [P] \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Therefore the computation of the optimal control function amounts to that of the predicate $[P]$. In this section we discuss how to compute $[P]$ for linear predicates P of the form: *

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \le b$$

or

$$AX \le b,\ A := [a_1, a_2, \ldots, a_n].$$

Denote by $\mathbf{G}_u$ the restriction of the model G to $\Sigma_u = \{\alpha_a, \alpha_2, \ldots, \alpha_m\}$. Let $E = [E_{\alpha_1}, E_{\alpha_2}, \ldots, E_{\alpha_m}]$. Then

$$\begin{aligned} [P] &= \{X \mid (\forall w \in L(\mathbf{G}_u,X))\ \delta(w,X) \in P\} \\ &= \{X \mid (\forall w \in L(\mathbf{G}_u,X))\ X + EV_w \in P\} \\ &= \{X \mid (\forall w \in L(\mathbf{G}_u,X))\ AX + AEV_w \le b\} \\ &= \{X \mid (AX + \max_{w \in L(\mathbf{G}_u,X)} AEV_w) \le b\} \end{aligned} \tag{4.2}$$

In general, $\max\limits_{w \in L(\mathbf{G}_u,X)} AEV_w$ is a difficult nonlinear programming problem with an unstructured feasible set $L(\mathbf{G}_u,X)$. However, the previous section has shown that when $\mathbf{G}_u$ is LF wrt Σ_u the feasible set $L(\mathbf{G}_u,X)$ can be characterized by a system of n linear inequalities. Based on this result the following theorem asserts that when $\mathbf{G}_u$ is LF wrt Σ_u the computation of $[P]$ is reduced to a linear integer programming problem.

Theorem 4.1

If $\mathbf{G}_u$ is LF then

$$[P] = \{X \mid AX + AEV^*(X) \le b\}$$

where $V^*(X)$ is the solution to the following linear integer programming problem:

$$\max BV \qquad s.t. \begin{cases} EV \ge -X \\ V \ge 0 \end{cases} \tag{4.3}$$

where $B := AE$.

□

From (4.2) and Theorem 4.1 we have

Corollary 4.1

The optimal control function f^* for a linear predicate $P := (AX \le b)$ can be computed as

$$f^*(\alpha,X) = \begin{cases} 1 & \text{if } AX' + BV^*(X') \le b,\ X' := X + E_\alpha \\ 0 & \text{otherwise} \end{cases}$$

---

* We only consider linear predicates specified by a scalar constraint; a linear predicate of vector form can be treated as the

□

We have seen that the linear integer programming problem (IP) is central to the computation of the optimal control function. IP has been extensively studied in the last two decades and many algorithms exist for its solution [8,9]. These algorithms are in general much more efficient than those based on state-graph search, which ignore the structure of the feasible solution set. Therefore, the computation of the optimal control function for VDES using IP can be substantially more efficient than that for unstructured models. Even though it has not been proved that these IP algorithms are of polynomial complexity -- many people doubt they are -- their tractability and efficiency have been demonstrated in practice. Our implementation, which uses the cutting-plane algorithm for the solution of IP, has also borne out this conclusion.

The integer programming problem (4.3) has m free variables and n constraints. In the rest of this section we show that for a given predicate P an IP equivalent to (4.3) but with fewer free variables and fewer constraints may be obtained, leading to further reduction of computation. Again this reduction is achieved by exploiting system structure. Indeed the following theorem asserts that when the structure matrices B and E are in appropriate forms such reduction is possible. Define an IP:

$$\max B_1 V_1 \tag{4.4}$$

$$s.t. \begin{cases} E_1 V_1 \geq -X_1 \\ V_1 \geq 0 \end{cases}$$

Let $B=[B_1,0]$ and $E=\begin{bmatrix} E_1 & E_2 \\ 0 & E_3 \end{bmatrix}$, $X=\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ with B_1, E_1, X_1 being matrices of dimensions $1{\times}m_1$, $n_1{\times}m_1$, $n_1{\times}1$ respectively, and 0 represents 0-matrices of appropriate dimensions.

Theorem 4.2

Assume $E_2 \leq 0$. Then the IPs (4.3) and (4.4) above have the same maximum, with $V^*(X) = [V_1^*(X_1),0]$.

□

Each column of E corresponds to an event and each row to a state variable; the submatrix E_1 of E is determined wholly by events $\alpha_1, \alpha_2, ..., \alpha_{m_1}$ and state variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n_1}$. Therefore if the structure matrices B and E are in the form specified above then only the subsystem consisting of events α_i $(i=1,2,...,m_1)$ and state variables $\mathbf{x}_i$ $(i=1,2,...,n_1)$ are relevant to the maximization.

Without loss of generality, let

$$\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n'}\} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n_1}\} \cup \{\mathbf{x}_i \mid a_i \neq 0\}.$$

Decompose the state space $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{Y} \oplus \overline{\mathbf{Y}},$$

where $\mathbf{Y} := \{[x_1, x_2, ..., x_{n'}] \mid x_i \in \mathbf{Z}\}$.

Theorem 4.3

If the matrices B and E are in the forms specified in Theorem 4.1 then $supC(P)$ is observable wrt $\mathbf{Y}$.

□

This theorem says that the optimal controller f^* enforcing $supC(P)$ only needs to observe the state variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n'}$; all others are irrelevant to the computation of the optimal control function.

Another special form of E which can lead to reduction of the programming problem is given by Theorem 4.4. For an IP:

---

conjunction of several scalar linear predicates and the corresponding optimal control function can be computed modularly [6,11].

$$\max B V \tag{4.5}$$

$$s.t. \begin{cases} E_1 V \geq -X_1 \\ V \geq 0 \end{cases}$$

let $E = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix}, X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$, with matrices E_1, X_1 of dimensions $n_1 \times m$, $n_1 \times 1$ respectively.

Theorem 4.4

Assume $E_2 \geq 0$. Then IPs (4.3) and (4.5) have the same optimal solutions, thus the same maximum.

□

In analogy to Theorem 4.3 we may conclude that, when E has the form specified in Theorem 4.4, **Y** is an observability subspace for $supC(P)$ and we only need to observe state variables $\mathbf{x}_i$ $(i=1,2,\ldots,n')$ in computing the optimal control function.

5. EXAMPLE: A PRODUCTION NETWORK

We consider the control of a production network consisting of two groups of machines, two buffers and a test unit. The Petri-net representation of the network is shown below

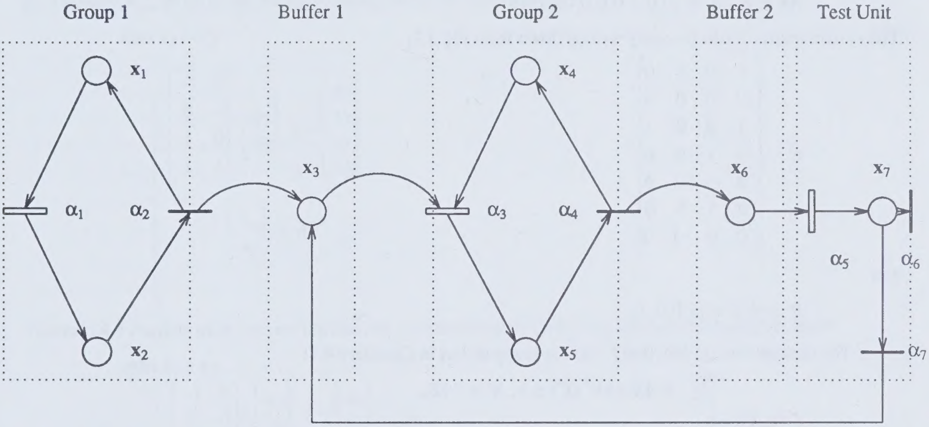

Group 1
Buffer 1
Group 2
Buffer 2
Test Unit
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\mathbf{x}_5$
$\mathbf{x}_6$
$\mathbf{x}_7$
α_1
α_2
α_3
α_4
α_5
α_6
α_7

Figure 5.1

α_1 (α_3): A machine in Group 1 (2) starts working

α_2 (α_4): A machine in Group 1 (2) finishes working

α_5: Test begins

α_6: Test is completed

α_7: Test unit rejects an imperfect part

$\mathbf{x}_1$ ($\mathbf{x}_4$): No. of idle machines in Group 1 (2)

$\mathbf{x}_2$ ($\mathbf{x}_5$): No. of working machines in Group 1 (2)

$\mathbf{x}_3$ ($\mathbf{x}_6$): Content of Buffer 1 (2)

$\mathbf{x}_7$: No. of parts under test

Assume that we can control only α_1, α_3, and α_5. Then $\Sigma_c = \{\alpha_1, \alpha_3, \alpha_5\}$, $\Sigma_u = \{\alpha_2, \alpha_4, \alpha_6, \alpha_7\}$. The uncontrollable part $\mathbf{G}_u$ of the system is obtained by removing α_1, α_3 and α_5 from Fig.5.1:

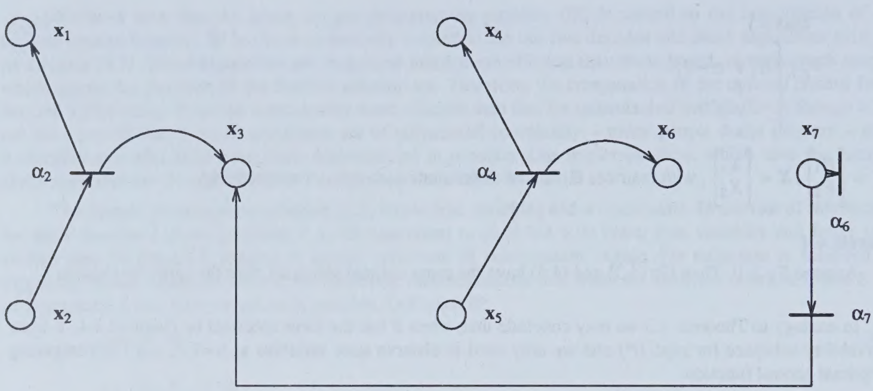


Figure 5.2

We consider the following legal-state specification:

$$P = (x_3 \leq b).$$

or,

$$AX \leq b, \; A := [0\;0\;1\;0\;0\;0\;0].$$

The system matrix E can be easily written down from Fig.5.2:

$$E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 \end{bmatrix}$$

and

$$B := A\,E = [1\;0\;0\;1].$$

The optimal control function f^* can be computed as in Corollary 4.1:

$$f^*(\alpha,X) = \begin{cases} 1 & \text{if } AX' + BV^*(X') \leq b, \; X' := X + E_\alpha \\ 0 & \text{otherwise} \end{cases}$$

where $V^*(X)$ is the solution to the following IP:

$$\max B\,V \tag{5.1}$$

$$s.t. \begin{cases} E\,V \geq -X \\ V \geq 0 \end{cases}$$

There are seven constraints and four free variables in the above IP. To simplify it we apply row and column interchanges to the matrices E and B to obtain ([7])

$$B = [1\;1\;0\;0]$$

$$E = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The four columns of the new matrix E correspond to events α_2, α_7, α_4, α_6 respectively; and the seven rows correspond to state variables x_2, x_7, x_1, x_3, x_4, x_5, x_6 respectively. The new matrices E and B have the forms:

$$B = [B_1, 0], \;\; E = \begin{bmatrix} E_1 & E_2 \\ 0 & E_3 \end{bmatrix}$$

with

$$B_1 = [1 \;\; 1]$$

$$E_1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \;\; E_2 = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \le 0, \;\; E_3 = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 1 & 0 \end{bmatrix}$$

By Theorem 4.2, IP (5.1) has the same maximum as IP (5.2) below:

$$\max u_1 + u_2 \tag{5.2}$$

$$s.t. \begin{cases} \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \ge - \begin{bmatrix} x_2 \\ x_7 \\ x_1 \\ x_3 \end{bmatrix} \\ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \ge 0 \end{cases}$$

Theorem 4.4 enables us to remove the last two constraints in IP (5.2), leading to further simplification:

$$\max u_1 + u_2 \tag{5.3}$$

$$s.t. \begin{cases} \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \ge - \begin{bmatrix} x_2 \\ x_7 \end{bmatrix} \\ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \ge 0 \end{cases}$$

Obviously, the maximum of the above IP is achieved when $u_1 = x_2$, $u_2 = x_7$. So

$$U^*(X) = \begin{bmatrix} x_2 \\ x_7 \end{bmatrix}.$$

and the optimal control function can be further written as:

$$f^*(\alpha, X) = \begin{cases} 1 \text{ if } X'(3) + X'(2) + X'(7) \le b, \; X' := X + E_\alpha \\ 0 \text{ otherwise} \end{cases} \tag{5.4}$$

Under the control of f^* the production network will have the maximal legal behavior, and to implement this controller we need to observe only state variables x_2, x_3, x_7.

6. CONCLUSION

One of the motivations for studying VDES is to find computationally efficient solutions to various control problems of DES by exploiting system structure. This paper provides such a solution. It is first shown that if a VDES has loop-free structure then the state set reachable from any initial state can be characterized by a system of linear inequalities. This result is used to show that when the uncontrollable part of the plant is loop-free, then the computation of the optimal control function enforcing a linear legal predicate is reduced to solving a standard linear integer programming problem. For an n-dimensional VDES with m uncontrollable events this integer programming problem has n constraints and m free variables; furthermore, under a suitable condition this programming problem can be reduced to another with fewer constraints and fewer free variables.

7. REFERENCES

[1] R. Karp and R. Miller, " Parallel program schemata, " *Journal of Computer and System Science* 3(4), 1969, pp. 167-195.

[2] R. Keller, " Formal verification of parallel programs, " *Communications of the ACM* **19**(7), 1976, pp 371-384.

[3] B.H. Krogh, " Controlled Petri nets and maximally permissive feedback logic, " *Proc. 25th Annual Allerton Conference*, September 1987.

[4] Y. Li and W.M. Wonham, " A state-variable approach to the modeling and control of discrete-event systems, " *Proc. 26th Annual Allerton Conference on Communication, Control, and Computing*, Univ. of Illinois at Urbana-Champaign, September 1988, pp.1140-1149.

[5] Y. Li and W.M. Wonham, " Controllability and observability in the state-feedback control of discrete-event systems," *Proc. 27th IEEE Conference on Decision and Control*, Austin, Texas, December 1988, pp.203-208.

[6] Y.Li and W.M. Wonham, " Composition and modular state-feedback control of vector discrete-event systems," *Proc. 23rd Annual Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, March 1989.

[7] Y.Li and W.M.Wonham, " Linear Integer Programming Techniques in the Control of Vector Discrete-Event Systems," Systems Control Group Report #8908, Dept. of Electrical Engineering, University of Toronto, June 1989.

[8] M. Minoux, *Mathematical Programming: Theory and Algorithms*, John Wiley & Sons , 1986.

[9] R.G. Parker and R.L. Rardin, *Discrete Optimization*, Academic Press, 1988.

[10] J. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, N.J., 1981.

[11] P.J. Ramadge and W.M. Wonham, " Supervisory control of a class of discrete event processes," *SIAM J.Control and Optimization*25(1), 1987, pp.206-230.

[12] P.J. Ramadge and W.M. Wonham, " Modular feedback logic for discrete event systems," *SIAM J. Control and Optimization* **25**(5), 1987, pp. 1202-1218.

[13] T. Ushio and R. Matsumoto, "State feedback and modular control synthesis in controlled Petri nets", *Proc. 27th IEEE Conference on Decision and Control*, Austin, Texas, December 1988, pp.1502-1507.

[14] W.M. Wonham, "A control theory for discrete-event systems," in *Advanced Computing Concepts and Techniques in Control Engineering* (M.J.Denham and A.J. Laub, Eds.), NATO ASI Series, vol. F47, Springer-Verlag, Berlin, 1988, pp.129-169.

Cooperative Subdivision Search Algorithms With Applications

N. Dadoun, D.G. Kirkpatrick
Department of Computer Science, University of British Columbia, Vancouver, BC, CANADA V6T 1W5

1. Introduction

We present parallel algorithms for a number of subdivision search and related problems which make use of a hierarchical representation. We concentrate on cooperative algorithms, algorithms for which multiple processors are employed to answer a single query. We discuss cooperative search in linear and planar subdivisions. Linear subdivision search is equivalent to sorted table lookup. Since the boundary of a convex polygon can be thought of as a linear subdivision, related applications include convex polygon separation and common/separating tangents. Since the surface of a convex polyhedron is topologically equivalent to a planar subdivision, related applications include convex polyhedron separation.

We assume a synchronous parallel model of computation in which all processors have access to a global shared memory which is used for all communication between processors. Our algorithms use the concurrent-read exclusive-write (CREW) rule for accessing the common memory.

We denote by n the number of vertices of a typical input object (subdivision, convex polygon or convex polyhedron). The number of processors available, k for $1 \le k \le n$, is an input parameter of our algorithms. Although our algorithms are applicable to the entire range of k, occasionally minor modifications are necessary to ensure correct behaviour for small values of k. When these modifications are straightforward, we omit their details in the interest of an uncluttered presentation.

The search algorithms presented are competitive with previously known algorithms which address the same problems and provide constant time solutions when a quasi-linear (n^{α} for some $\alpha > 0$) number of processors is available. They also have the interesting property that as the number of processors available is reduced to 1, the time bounds increase smoothly to provide sequential algorithms which match the best sequential times known for the same problems.

We present algorithms for linear subdivision search (and subsequently, convex polygon tangent and convex polygon separation algorithms) which run in $\Theta(\log n/(1 + \log k))$ time. We also present planar subdivision search algorithms which, with suitable preprocessing, answer point location queries in $O(\log n/(1 + \log k))$ time. This in turn leads to convex polyhedron/linear subspace separation algorithms which run in $O(\log n/(1 + \log k))$ time and convex polyhedron/polyhedron separation algorithms which run in $O((\log n/(1 + \log k))^2)$ time.

2. Linear Subdivision Search and Polygon Hierarchies

A linear subdivision is a partitioning of the real line into contiguous segments. Given a real value, the linear subdivision search problem involves the identification of the segment which contains that value. Recall that the sorted table look-up problem is defined as follows: given a sorted table of n real numbers $x_1, x_2, \dots x_n$ and a search key x_s, return the index i such that $x_s \in [x_i, x_{i+1}]$ (with $x_0 = -\infty$ and $x_{n+1} = \infty$). Clearly, the linear subdivision search problem is equivalent to the sorted table lookup problem.

Snir [S] has shown that in the CREW model of computation, with k processors available ($1 \le k \le n$), the sorted table look-up problem has a time bound of $\Theta(\log n/(1 + \log k))$. In the absence of concurrent read, the sorted table look-up problem has time bound $\Theta(\log n - \log k)$. (This is a key result which demonstrates that processors with a concurrent read facility are strictly more powerful than processors without concurrent reads.)

Previous work has provided logarithmic time sequential algorithms for common tangents [OvL] and convex polygon separation [E]. Viewing the boundary of each polygon as a sequence of values, both of these algorithms involve a 'binary search' approach; within each iteration, the midpoints of the remaining sequence of vertices of each polygon are compared. Then, a case analysis is applied to reduce at least one of the polygons by a half. The iterations continue until both polygons are reduced to a constant size. Therefore, there are a logarithmic number of constant time iterations. Although these algorithms do not seem to admit to straightforward parallel implementations in our model, their binary search approach forms the basis for the k-way search algorithms developed here.

Chazelle and Dobkin [CD] and Dobkin and Kirkpatrick [DK1][DK2][DK3][DK4] have presented sequential algorithms for a number of convex polygon and polyhedron separation problems. Some of the results presented here can be viewed as non-trivial parallel generalizations of their work.

The natural extensions of the algorithms presented by Atallah and Goodrich [AG1] for n-vertex convex polygon common tangents and separation run in $O(\log^2 n/(1 + \log^2 k))$ time using k CREW processors. Therefore, although they provide constant time algorithms (with a sublinear number of processors) for the problems which they address, they do not achieve the theoretical maximum speed-up of log k.

The results reported here unify and extend the work cited above. Hereafter, all polygons considered are convex. We assume a very natural presentation of the input; the vertices of a convex polygon are provided in an array in clockwise order of their appearance on the boundary. As exploited by [DK1], this representation contains enough structure to provide the hierarchy of polygons used in our algorithms without preprocessing.

By noting that any subsequence of P's vertices defines another convex polygon completely contained within P, we can recursively define an *(inner) polygonal hierarchy* of P. Our algorithms make use of a *polygon approximation*: any collection of 3 or more vertices taken in (clockwise) order from a convex polygon forms an approximation to P which we will refer to as a *subpolygon* and denote by $\tilde{P}$. The k vertices ($3 \leq k \leq n$) of $\tilde{P}$ are subscripted in clockwise order with p_{i-1} and p_{i+1} respectively preceding and following p_i. A sequence of k evenly spaced vertices in P is a k–*subpolygon*; this can be considered a canonical subpolygon approximation attuned to the value k. When the vertices of P are presented in an array, all such canonical representations of P are available by indexing. We use P[a, b] to denote the range of points of P clockwise from a to b inclusive; this will be abbreviated to [a, b] when the context is clear.

Given a subpolygon $\tilde{P}$, convexity imposes implicit restrictions on how far P can extend beyond the edges of $\tilde{P}$. Some observations will clarify how the convexity restrictions and the polygon hierarchies are used together: Consider two non-intersecting chords ab and cd of convex polygon P. Consider the two half-planes H^+ and H^- (respectively, K^+ and K^-) defined by line ab (respectively, line cd).

Observation 2.1: The entire sequence of vertices of P between a and b lie in H^- and the entire sequence of vertices of P between b and a lie in H^+.

Observation 2.2: One of the regions (say, $H^- \cap K^-$) cannot contain any vertices of P in its interior. Furthermore, the entire sequence of vertices between d and a (and between b and c) lies in $H^+ \cap K^+$.

Applying these observations about chords of P to the edges of its subpolygon $\tilde{P}$ implicitly defines $\hat{P}$: the star-shaped (possibly unbounded) polygon which is the union of all possible convex polygons which could have $\tilde{P}$ as a subpolygon.

The key to the efficient parallel use of the subpolygon approximations is the judicious assignment of processors. Intuitively, this can be thought of as a direct analogue of parallel search in a sorted table; the k-way search corresponds to jumping through subpolygons which form an implicit inner polygon hierarchy.

This correspondence allows the design of k-processor convex polygon algorithms which are natural extensions of k-way search. Within each iteration of the k-way search algorithm for the sorted table lookup problem, the search key's containment in a particular subsequence can be determined by a single processor in constant time. This is slightly more involved for the common tangents/ separation problems but can be done efficiently by exploiting convexity and hierarchical properties of a polygon approximation.

Within each iteration of the algorithms for common tangents and separation, an approximate solution is found and then a case analysis is applied to reduce the region of interest. This can be thought of as 'electing' an active processor (the one corresponding to the subpolygon/subsequence solution) which can then apply the case reduction. It then writes the reduced ranges into a predetermined location and the remaining processors use the concurrent read facility to update their information. Since only one processor is assigned to the pair realizing the subpolygon solution, this election process avoids write contention.

In the next section, we address the point in polygon and polygon separation problems; in the final paper [DaK3], we also address polygon common/separating tangents.

3. Linear Subdivision Search Application: Polygon Separation

Determining the separation of geometric objects is a generalization of detecting their intersection; it is the identification of a witness to the minimum distance separating them. If the objects do intersect the separation is zero and the witness is a point in their intersection.

Consider a special case of planar subdivision search: Given an n-vertex convex polygon P and a point q, determine whether or not q lies inside P. Ignoring degeneracies, this question can be answered by placing a horizontal line L through q, finding the (0 or 2) intersections of L with P, and testing q's position with respect to the intersections. By decomposing P into two chains which are monotone with respect to the y-axis, we can use a natural extension of linear subdivision search to answer the line/polygon intersection (hence point-in-polygon) question in $O(\log n/(1 + \log k))$ time using k processors.

We first consider the problem of finding the separation of an n-vertex convex polygon P and a line segment r outside P. Given $\tilde{P}$, a subpolygon approximation of P, and a vertex p_i of $\tilde{P}$, in constant time a single processor can determine whether or not the point (not necessarily a vertex) realizing the separation between $\tilde{P}$ and r lies in $[p_{i-1}, p_{i+1}]$ by determining the distances between r and (i) the vertex p_i, (ii) the segment $p_{i-1}p_i$, and (iii) the segment p_ip_{i+1}. If the point realizing the separation between $\tilde{P}$ and r is vertex p_i, then by Observation 2.2, the separation of r and P must be realized by a point in $[p_{i-1}, p_{i+1}]$. If the separation between $\tilde{P}$ and r is realized by an interior point of segment $p_{i-1}p_i$ (respectively, segment p_ip_{i+1}), then by Observation 2.1, the separation of r and P must be realized by a point in $[p_{i-1}, p_i]$ (respectively, $[p_i, p_{i+1}]$). Otherwise, the separation is not realized by a point in $[p_{i-1}, p_{i+1}]$.

This suggests the following k processor algorithm for the segment/polygon separation problem, which is essentially the algorithm presented by Atallah and Goodrich [AG1]: Form the p–subpolygon $\tilde{P}$ from P and repeat the following until $\tilde{P}$ is reduced to a single point. Assign a processor to each vertex of $\tilde{P}$ and determine which vertex or segment contains a point realizing the separation of segment r and $\tilde{P}$ in constant time. By the argument above, this identifies an index i such that the separation of r and P must lie between p_{i-1} and p_{i+1}. Replace $\tilde{P}$ with the p-subpolygon of $[p_{i-1}, p_{i+1}]$.

Each iteration of this procedure reduces the length of the sequence of edges under consideration by a factor of k while maintaining the invariant that the remaining range of edges contains a point which realizes the desired separation. Therefore, the segment/polygon separation can be found in $O(\log n/(1 + \log k))$ time. This algorithm can be used to find line/polygon separation by regarding the line as a segment with its endpoints extended to infinity. Similarly, this algorithm can be used to find point/polygon separation by regarding the point as a degenerate segment with zero length.[1] This last application can be thought of as a table-lookup routine: finding the minimum in a virtual table of distances from a fixed point. Note as well that this algorithm can be adapted to find the extreme vertex of a convex polygon in any given direction within the same resource bounds.

We turn to the problem of finding the separation of n-vertex convex polygons P and Q. The polygon/polygon separation algorithm uses a $\sqrt{k}$-subdivision technique [ACGOY][KR][AG1] to achieve an optimal algorithm. Within each iteration the separation between the current $\sqrt{k}$-subpolygons is determined and a case analysis is employed to reduce one of the subpolygons under consideration to a constant number of edges.

---

[1] Note that due to the non-unimodality of distances in a convex polygon [E], this technique cannot be used directly for the farthest point problem.

The polygon/polygon separation case analysis provided in Edelsbrunner [E] produces a sequential O(log n) time separation algorithm. If concurrent write is available, it is possible to use Edelsbrunner's case analysis to design a polygon/polygon separation algorithm using a $\sqrt{k}$-subdivision technique which runs in O(log n/(1 + log k)) time using k CRCW processors. In the absence of a concurrent write facility, it is not straightforward to coordinate the $\sqrt{k}$ case results at each vertex of the subpolygons under consideration. Our algorithm avoids this coordination problem by "electing" the processor which corresponds to the subpolygon minimum separation to apply the case analysis.

Note that polygon separation is realized by points on the polygon boundary, therefore we can restrict our attention to chain-chain separation. We maintain a subpolygon as a *monotone polygonal sector* (MPS)[DK1]: The (convex) area delimited by a monotone sequence of vertices and edges in order of increasing y-coordinate augmented with semi-infinite rays at the beginning and end of each chain. A convex polygon P can be represented as the intersection of a right-oriented MPS P_R and a left-oriented MPS P_L. Separation questions for P can be reduced to separation questions for P_R and P_L. The algorithm described above to find the extreme vertices of a convex polygon can be used to preprocess P into P_R and P_L in O(log n/(1 + log k)) time using k CREW processors.

We state without proof several useful properties of MPSs: Two convex polygons P and Q intersect iff P_L intersects Q_R and Q_L intersects P_R. Ignoring degeneracies, two MPSs oriented in opposite directions can intersect in either 0 or 2 points. If P and Q do not intersect then the separation between P and Q will be realized either by the separation between P_L and Q_R or the separation between Q_L and P_R.

These properties simplify several aspects of determining polygon/polygon separation. If the polygons P and Q do intersect, then their corresponding (oppositely-oriented) MPS boundaries have a constant number of intersections and and a witness to the intersection of P and Q can be easily determined.

The notion of subpolygon extends in the natural way to an MPS. Hereafter, we maintain subpolygons $\tilde{P}_L$ and $\tilde{Q}_R$ with their vertices indexed by increasing y-coordinate as $\tilde{P}_L = [p_{lower\_P}, p_{upper\_P}]$ and $\tilde{Q}_R = [q_{lower\_Q}, q_{upper\_Q}]$. Initially, the lower and upper bounds correspond to minimum and maximum y-coordinates respectively. Within each iteration of the algorithm, the bounds are adjusted as the range of points which could realize the separation is reduced; we maintain the invariant that the current MPSs $[p_{lower\_P}, p_{upper\_P}]$ and $[q_{lower\_Q}, q_{upper\_Q}]$ contain either a pair of points realizing the minimum distance separating P_L and Q_R or a point in the intersection of P_L and Q_R. Thus, the algorithm can be viewed as a search routine to identify p^* and q^*, points of P_L and Q_R respectively, which realize the MPS separation.

In general, the separation will be realized by a vertex-vertex, vertex-segment, or segment-segment pair. We will not consider the segment-segment configuration; the algorithm can be easily modified to handle it. We maintain the segment T with endpoints p_i and q_j defining the separation between subpolygons $\tilde{P}_L$ and $\tilde{Q}_R$. This has the obvious interpretation in the vertex-vertex case; for ease of exposition in the vertex-segment case, we assume the existence of a pseudo-vertex on the segment in question which realizes the separation and assume that the vertices of the polygon have been relabeled to incorporate it into the vertex ordering.

Given $\tilde{P}_L$ and $\tilde{Q}_R$, MPS subpolygons of P and Q respectively, and the points p_i and q_j which realize the separation of $\tilde{P}_L$ and $\tilde{Q}_R$, the following two lemmas describe how to localize the points under consideration for realizing the separation of P and Q.

Lemma 3.1 (The same side lemma): Given MPS subpolygons $\tilde{P}_L = [p_{lower\_P}, p_{upper\_P}]$ and $\tilde{Q}_R = [q_{lower\_Q}, q_{upper\_Q}]$ and vertices p_i and q_j realizing their separation then

(i) either $p^* \in [p_{lower\_P}, p_{i+2}]$ or $q^* \in [q_{lower\_Q}, q_{j+2}]$

and (ii) either $p^* \in [p_{i-2}, p_{upper\_P}]$ or $q^* \in [q_{j-2}, q_{upper\_Q}]$.
Furthermore, this can be determined by a single processor in constant time.

Proof: It is fairly straightforward to demonstrate that (i) and (ii) are true. The difficulty lies in the actual determination of which portions of which MPSs can be discarded. This can be accomplished via a case analysis which is included in the final paper [DaK3]. ♦

In order to properly reduce the number of vertices under consideration, it is necessary to ensure that one of $\tilde{P}_L$ and $\tilde{Q}_R$ be reduced to a constant number of segments. Note that this is not guaranteed by Lemma 3.1. It is possible, for example, that only the upper portion of $\tilde{P}_L$ and the lower portion of $\tilde{Q}_R$ are eliminated. The following Lemma shows that a full reduction is always possible for at least one of the subpolygon approximations.

Lemma 3.2 (The kitty-corner lemma): Given MPS subpolygons $\tilde{P}_L = [p_{lower\_P}, p_{upper\_P}]$ and $\tilde{Q}_R = [q_{lower\_Q}, q_{upper\_Q}]$ and vertices p_i and q_j realizing their separation then

(i) either $p^* \in [p_{i-2}, p_{upper\_P}]$ or $q^* \in [q_{lower\_Q}, q_{j+2}]$
and (ii) either $p^* \in [p_{lower\_P}, p_{i+2}]$ or $q^* \in [q_{j-2}, q_{upper\_Q}]$.

Furthermore, this can be determined by a single processor in constant time.

Proof: As in Lemma 3.1, it is fairly straightforward to demonstrate that (i) and (ii) are true. The complete case analysis is included in the final paper [DaK3]. ♦

Theorem 3.1: The separation of two convex n-vertex polygons P and Q can be determined in O(log n/(1 + log k)) time using k CREW processors.

Proof: Since the polygons P and Q are presented as ordered arrays of vertices, the subpolygon approximations are implicit and immediately available through indexing. Within each iteration, the vertices realizing the separation of $\tilde{P}_L$ and $\tilde{Q}_R$ are determined in constant time. By Lemmas 3.1 and 3.2, each iteration takes constant time and the candidate number of vertices on at least one of the polygons can be reduced by a factor of $\sqrt{k}$ during each iteration. Therefore, there are a total of O(log n/(1 + log k)) iterations. ♦

The final paper [DaK3] presents a similar argument to produce equivalent results for the construction of separating and common tangents:

Theorem 3.2: The separating or common tangents of two convex n-vertex polygons P and Q can be constructed in O(log n/(1 + log k)) time using k CREW processors.

As well, the final paper [DaK3] includes an implicit reduction of the sorted table-lookup problem to convex polygon common/separating tangents and separation. This result, Theorems 3.1 and 3.2 above and the lower bounds on the sorted table-lookup problem demonstrated by Snir [S] combine to yield:

Corollary 3.1: Finding the separating or common tangents of a vertex with an n-vertex convex polygon with k CREW processors has time complexity $\Theta(\log n/(1+\log k))$.

Corollary 3.2: Finding the separation of a vertex with an n-vertex convex polygon with k CREW processors has time complexity $\Theta(\log n/(1+\log k))$.

4. Cooperative Planar Subdivision Search

Previous research into parallel algorithms for subdivision search has concentrated on parallel preprocessing of a planar subdivision to facilitate sequential subdivision search. Our cooperative search algorithms use extensions of the subdivision hierarchy presented by Kirkpatrick [K].

We use the term *ordinary hierarchy* to refer to an unaugmented subdivision hierarchy. The standard doubly connected edge list representation can be used to implement each of the subdivisions in the ordinary hierarchy.

Dadoun and Kirkpatrick [DaK1] [DaK2] show that given a planar subdivision on n vertices, a hierarchical representation with O(log n) elements can be constructed in O(log n) expected parallel time using a randomized algorithm or O(log n $\log^* n$) parallel time using a deterministic algorithm. Cole and Zajicek [CZ] describe how to construct a subdivision hierarchy in O(log n) parallel time using O(n/log n) processors. However, their construction makes use of the duration unknown task scheduling (DUTS) algorithm which uses expander graphs and thus has rather large constant factors.[1]

As has been discussed elsewhere [DK4][DaK1] [DaK2], the ordinary subdivision hierarchy when interpreted as a polyhedral hierarchy can be used to answer many different intersection and separation queries involving polyhedra. Convex polyhedron separation queries with points, lines and planes can be answered in O(log n) sequential time.

Exploiting a subdivision hierarchy for cooperative search algorithms is not as simple as the linear subdivision search or, equivalently, the use of polygonal hierarchies in 2 dimensions. The hierarchy is not implicit within the representation; the hierarchy must be explicitly constructed within a preprocessing phase. Additional pointer information must be incorporated to allow "jumping through" levels of the hierarchy; this was available within the 2–dimensional structure simply by using indexing. Localizing the search within the hierarchy by finding descendants of a particular element is more complicated; this could be done within the linear subdivision (or polygonal hierarchy) by limiting the index range under consideration. However, when properly augmented, the subdivision hierarchy can be used for cooperative subdivision search queries. It is worth noting that attempting to construct an augmented hierarchy in the obvious way – by collapsing multiple levels of the ordinary hierarchy – will introduce an unacceptable space cost.

Iterated application of the Lipton/Tarjan separator theorem can be used to construct a space-efficient augmented hierarchy for cooperative point location by k processors. In this *static hierarchy*, the value of k is built directly into the data structure.

Theorem 4.1 [LT2]: Let G be an n-vertex planar graph with nonnegative vertex costs summing to no more than one. Given ε such that $0 < \varepsilon \le 1$, there is some set C of $O(\sqrt{n/\varepsilon})$ vertices whose removal leaves G with no connected component of cost exceeding ε.

As described by Lipton and Tarjan [LT2], this set C is identified through iterated application of their planar separator theorem [LT1]. The parallel simple cycle separator algorithm due to Miller [M] can be employed to identify an equivalent set C in polylogarithmic time using a polynomial number of processors.

Theorem 4.2: Given a planar subdivision G on n vertices and an integer k for $1 \le k \le n$, it is possible to construct an O(n)-space data structure which will allow k processors to perform a point-location query in $O(\log n / (1 + \log k))$ parallel time.

Proof: We assume the subdivision is triangulated. By Theorem 4.1, for a given ε and vertex cost assignment summing to less than one, a small separator C can be identified whose removal will leave no component of size exceeding ε.

This can be used to construct a hierarchy of subdivisions which can be used for cooperative point location by k processors. Each subdivision in the hierarchy is derived from its predecessor by removing the isolated components and retriangulating the separator.

It is necessary to control the size of the triangulation which results from the removal of a component to ensure that any component can be searched by k processors in constant time. This requires relating the weight of a component to the "size of the hole" remaining when it is removed. This can be achieved by assigning each vertex v the weight: degree(v)/(6n – 12).

The static hierarchy is customized for a given k, therefore it is necessary to address the structure of the hierarchy for small values of k. Appealing to Theorem 4.1, if $\varepsilon \le 1/n$, $|C| = n$ and the theorem holds trivially but no components can be removed. Since $|C| = O(\sqrt{n/\varepsilon})$, there exist constants c and n_0 such that for all $n \geq n_0$, $|C| \leq c\sqrt{n/\varepsilon}$. We are interested in values of ε for which $c\sqrt{n/\varepsilon} < n$. Therefore, $\varepsilon > c^2/n$. (Lipton and Tarjan's separator theorem [LT1] gives $c < 16$; Venkatesan [V] improves this to $c < 5$.)

---

[1] We have recently learned of the work of Tamassia and Vitter [TV] who provide a parallel construction for the planar point location structure of Edelsbrunner, Guibas and Stolfi [EGS]. It is not clear whether or not that structure can be augmented for cooperative search or used for the polyhedral applications discussed here.

Therefore, Theorem 4.1 can be used to construct a static hierarchy for k such that $1 \leq k \leq n$ by choosing

$$\varepsilon = \begin{cases} c^2/n & \text{if } 1 \leq k \leq c^2 \\ k/n & \text{if } c^2 < k \leq n. \end{cases}$$

Since c^2 is a constant, each component can still be searched by a single processor in constant time.

Using the algorithm associated with Theorem 4.1 on G with the given weight scheme on the vertices and the choice of ε specified above, it is possible to identify a separating set C of size $O(n/\sqrt{k})$ such that the remaining connected components have size O(k). Removing the connected components and retriangulating produces a new planar subdivision G_1 of size $O(n/\sqrt{k})$.

This entire process can be repeated to produce a sequence of m subdivisions, $G=G_0$, G_1 to G_m such that the size of G_m is O(k). Since the size of G_0 is n and each subdivision is a $\sqrt{k}$ factor smaller than its predecessor, m is $\log_{\sqrt{k}}(n)$ or $O(\log n / (1 + \log k))$.

Consider the space requirement of this static hierarchy. Each facet f which is removed from subdivision G_i keeps one pointer to the list of facets in subdivision G_{i+1} which resulted when f's component was removed and retriangulated. Thus, each component of size O(k) which is removed within a subdivision G_i keeps one list of the faces in its corresponding retriangulation in G_{i+1}. This "sharing of pointers" from level to level only requires a linear number of pointers to be associated with a given level. Since the size of the levels is geometrically decreasing, the overall storage is linear.

The associated search algorithm is a simple extension of the search algorithm associated with the ordinary subdivision hierarchy. The search begins at the last level G_m. Within an iteration, the k processors search the O(k) target faces in the current level G_i in constant time. A unique "winner" is determined which writes its index into a predetermined location. The k processors use the concurrent read facility to determine which processor was successful and reallocate themselves to the associated sequence of O(k) faces in level G_{i-1}. This continues until the query point is located in $G_0 = G$. A constant amount of time is required for each level, and therefore the overall query time is the number of levels, $O(\log n / (1 + \log k))$. ♦

The static hierarchy constructed using the algorithm associated with Theorem 4.1 can be considered a straight-forward generalization of the subdivision hierarchy. In the ordinary hierarchy, the removable component weights are held to be constant and the components to be removed are restricted to being single vertices. The fact that an independent set of low-degree vertices which forms a fixed fraction of the size of the subdivision can be identified quickly is used in place of the (more complicated) separator theorem. There is an interesting dichotomy between the two hierarchies: the ordinary hierarchy is built bottom up by identifying the components to be removed; the static hierarchy is built top-down by identifying the vertices in the separator. Note that this is a simpler application of Lipton and Tarjan's separator result to subdivision search than that originally presented in [LT2].

The next issue to address is the existence of a space-efficient *dynamic hierarchy* structure – one which will admit to $O(\log n/(1 + \log k))$-time k-processor point location for all values of k such that $1 \leq k \leq n$ and only requires linear space.

One way of building a (space-inefficient) structure is to simply construct separate static hierarchies for $k = 2^i$ for $0 \leq i \leq \log(n)$[1]. This structure consists of log(n) static hierarchies, each of which requires O(n) space, resulting in an O(nlog(n)) space requirement overall. Given a value of k in the valid range, one of the available static hierarchies will be within a constant factor (2) of k.

---

[1] For simplicity in this discussion, we ignore problems in dealing with small values of k.

This first attempt at a dynamic hierarchy can be improved by noting that the speed-up provided by a static hierarchy is in log(k). Therefore, it would suffice to construct separate static hierarchies for $k = 2^{2^i}$ for $1 \leq i \leq \text{loglog}(n)$ resulting in an O(nloglog(n))-space structure. Given a value of k in the valid range, we could use the static hierarchy for the largest $2^{2^i} \leq k$ and still achieve O(log n/(1 + log k))-time k-processor point location.

In both of the attempts at a dynamic hierarchy above, the space cost of each of the component static hierarchies is dominated by its largest subdivision; in all cases above, the original subdivision. A space-efficient structure can be constructed by using static hierarchies to augment an ordinary hierarchy. In this scheme, the static hierarchies are 'rooted' at different subdivisions within the ordinary hierarchy:

Theorem 4.3: Given a planar subdivision G on n vertices, it is possible to construct an O(n)-space data structure which will allow k processors to perform a point-location query in O(log n / (1 + log k)) parallel time for any integer $1 \leq k \leq n$.

Proof: Recall that an ordinary hierarchy for a planar subdivision G on n vertices is a sequence of w subdivisions, $G=G_0$, G_1 to G_w such that for constants α and β (i) $|G_{i+1}| = |G_i| / \alpha$ for some $\alpha > 1$, (ii) $|G_w|$ is O(1) and therefore $w = \log_\alpha n$, and (iii) each region of G_{i+1} overlaps at most β regions in G_i. The *dynamic augmented hierarchy* is an ordinary subdivision hierarchy which has been augmented to allow different entry points for different specific values of k. The larger the value of k, the 'higher' the entry point within the ordinary hierarchy.

For simplicity, we assume that $n = 2^{2^t}$ (and therefore, t = loglog n) for some integer t. We assume that we have an ordinary subdivision hierarchy constructed for G as described above. We augment each of the elements $G_{2^{t-i}}$ for $1 \leq i \leq \text{loglog}(n)$ with a static hierarchy for $k = 2^{2^i}$.

Consider performing a point location query with k processors such that $1 \leq k \leq n$. We can assume that $k = 2^{2^i}$ for $1 \leq i \leq \text{loglog}(n)$, otherwise as described above we can use the largest $2^{2^i} \leq k$. Our structure should enable k processors to answer a point location query in G in O(log n / (1 + log k)) = $O(2^{t-i})$ steps. Therefore, we use the k processors to perform a cooperative search query using the static hierarchy associated with element $G_{2^{t-i}}$ in $O(2^{t-i})$ steps. Given the query point's location in $G_{2^{t-i}}$, a single processor can employ the ordinary hierarchy to identify the face containing the query point in another 2^{t-i} steps, each of which require constant time. Therefore, this dynamic hierarchy enables k processors to perform a point location query in G in O(log n / (1 + log k)) time.

We show that this dynamic hierarchy uses O(n) space. The ordinary hierarchy only requires O(n) space. Denote the size of the augmented structure associated with ordinary hierarchy element $G_{2^{t-i}}$ by $S_{2^{t-i}}$. Let $m = | G_{2^{t-i}} | = n / \alpha^{2^{t-i}}$ for $\alpha > 1$ such that α is the reduction factor associated with the ordinary hierarchy. Therefore,

$$\begin{aligned} S_{2^{t-i}} &\leq m + \frac{m}{\sqrt{k}} + \frac{m}{(\sqrt{k}\,)^2} + \frac{m}{(\sqrt{k}\,)^3} + \;\ldots \\ &= \frac{m}{1 - \frac{1}{\sqrt{k}}} = \frac{m\sqrt{k}}{\sqrt{k} - 1} \\ &= m\,\frac{2^{2^{i-1}}}{2^{2^{i-1}} - 1} \\ &\leq \frac{2n}{\alpha^{2^{t-i}}}. \end{aligned}$$

Therefore, the total size contribution of the augmented static hierarchies forms a constant multiple of a subsequence of the ordinary hierarchy's geometric sequence. Therefore, the overall space requirement for the dynamic augmented hierarchy is O(n). ♦

Note that if α were greater than or equal to 2 then it would not be necessary to augment the ordinary hierarchy; the space reduction from level to level would be sufficient to allow k processors to search the ordinary hierarchy element $G_{2^{t-i}}$ directly in constant time. Typically, α is much closer to 1, necessitating the use of the augmenting static hierarchies.

With regard to construction times for an n-vertex subdivision, the ordinary hierarchy can be constructed sequentially in O(n) time [K] and in parallel in O(log n) time using O(n/log n) processors with an expander graph construction [CZ] or in O(log n log*n) time using O(n/(log n log*n)) processors by identifying fractional independent sets [DK].

The static augmented hierarchy (for a given k) can be constructed sequentially in O(n log n) time [LT2] [LT3] [V] or in parallel in $O(\log^2 n)$ time using n^3 processors employing iterated application of Miller's separator algorithm [M]. The dynamic augmented hierarchy incorporates an ordinary hierarchy and a geometrically decreasing sequence of static hierarchies resulting in overall O(n log n) sequential time construction or $O(\log^2 n)$ parallel time construction using n^3 processors. The initial triangulation and subsequent retriangulations can be performed within these resource bounds [G].

5. Applications: Polyhedral Separation

As in [DK3][DK4][DaK1] and [DaK2], we exploit the fact that the surface of a convex polyhedron is topologically equivalent to a bounded planar subdivision and define an hierarchical representation for convex polytopes similar to the hierarchical representation for planar subdivisions [DK4].

With the augmented polyhedral hierarchy in place, cooperative polyhedron separation algorithms follow from their sequential equivalents in [DK3] and [DK4] and the cooperative search algorithm outlined in Section 4.

Theorem 5.1: Given an n-vertex convex polyhedron P which has been preprocessed into a dynamic augmented polyhedral hierarchy and a direction d, the extremum of P in direction d can be determined in O(log n/(1 + log k)) time using k processors.

Theorem 5.2: Given a linear subspace L and an n-vertex convex polyhedron P which has been preprocessed into a dynamic augmented polyhedral hierarchy, the separation of P and L can be determined in O(log n/(1 + log k)) time using k processors.

Theorem 5.3: Given two n-vertex convex polyhedra P and Q which have been preprocessed into dynamic augmented polyhedral hierarchies, the separation of P and Q can be determined in $O((\log n/(1 + \log k))^2)$ time using k processors.

The polyhedron extremum and polyhedron/linear subspace separation results are straightforward generalizations of the point location algorithms. The polyhedron/polyhedron separation result uses the $\sqrt{k}$-subdivision technique which was employed in the polygon separation algorithm of Section 3. The $\sqrt{k}$-hierarchies are used for each of P and Q; the k processors available are used to initialize the parallel planes of support which delimit the separation of the smallest ($\sqrt{k}$ facet) element of each of P's and Q's $\sqrt{k}$-static hierarchy. The algorithm proceeds by taking a step in one of the hierarchies (say P's) and using Q's hierarchy to update the separation. This continues until the top of both hierarchies has been reached and the separation of P and Q has been determined. Complete implementation details will be provided in the final paper [DaK4].

References

[ACGOY] Aggarwal, A., Chazelle, B., Guibas, L., O'Dunlaing, C., and Yap, C., "Parallel computational geometry", *Algorithmica* 3, 1988, pp. 293-327.

[AG1] Atallah, M.J. and Goodrich, M.T., "Parallel algorithms for some functions of two convex polygons", *Algorithmica* 3, 1988, pp. 535-548.

[AG2] Atallah, M.J. and Goodrich, M.T., "Efficient parallel solutions to some geometric problems", *Journal of Parallel and Distributed Computing*, 1986, pp. 492-507.

[CD] Chazelle, B. and Dobkin, D., "Detection is easier than computation", *ACM Symp. on Theory of Computing*, 1980, 146-153.

[C] Cole, R., "Parallel merge sort", *Proc. 27th Annual IEEE Symp. on Foundations of Comp. Sci.*, 1986, pp. 511-516.

[CZ] Cole, R. and Zajicek, O., "An optimal parallel algorithm for building a data structure for planar point location", To Appear in *Journal of Parallel and Distributed Computing*.

[DaK1] Dadoun, N. and Kirkpatrick, D.G. "Parallel processing for efficient subdivision search", *Proc. of the 3rd ACM Symposium on Computational Geometry*, 1987, pp. 204-214.

[DaK2] Dadoun, N. and Kirkpatrick, D.G., "Parallel construction of subdivision hierarchies", To Appear in *Journal of Parallel and Distributed Computing.*

[DaK3] Dadoun, N. and Kirkpatrick, D.G., "Optimal parallel algorithms for convex polygon separation", Computer Science Department TR 89-21, University of British Columbia, Sept. 1989.

[DaK4] Dadoun, N. and Kirkpatrick, D.G., "Cooperative algorithms for planar point location and convex polyhedron separation", In preparation, 1989.

[DK1] Dobkin, D.P. and Kirkpatrick, D.G., "Fast detection of polyhedral intersections", *Proceedings of the International Colloquium on Automata, Languages and Programming*, 1982, pp. 154-165.

[DK2] Dobkin, D.P. and Kirkpatrick, D.G., "Fast detection of polyhedral intersection", *Theoretical Computer Science* 27, 1983, pp. 241-253.

[DK3] Dobkin, D.P. and Kirkpatrick, D.G., "A linear time algorithm for determining the separation of convex polyhedra", *Journal of Algorithms* 6, 3, 1985, pp. 381-392.

[DK4] Dobkin, D.P. and Kirkpatrick, D.G., "Fast algorithms for preprocessed polyhedral intersection detection and related problems", In Preparation.

[E] Edelsbrunner, H., "Computing the extreme distances between two convex polygons", *Journal of Algorithms* 6, 1985, pp. 213-224.

[EGS] Edelsbrunner, H., Guibas, L.J, and Stolfi, J. "Optimal point location in a monotone subdivision", *SIAM Journal of Computing* 15, 1986, pp. 317-340.

[G] Goodrich, M.T. "Triangulating a polygon in parallel", To appear, *Journal of Algorithms.*

[KR] Karp, R.M., and Ramachandran, V. "A survey of parallel algorithms for shared-memory machines", Report No. UCB/CSD 88/408, Comp. Sci. Division, UC Berkeley, March, 1988.

[K] Kirkpatrick, D.G., "Optimal search in planar subdivisions", *SIAM Journal of Computing* 12,1, January 1983, pp. 28-35.

[LT1] Lipton, and Tarjan, "A separator theorem for planar graphs", *SIAM Journal of Applied Math.*, 36, 2, April 1979, pp. 177-189.

[LT2] Lipton, and Tarjan, "Applications of a planar separator theorem", *Proc. 18th Annual IEEE Symp. on Foundations of Comp. Sci.*, 1977, pp. 162-170.

[M] Miller, G.L., "Finding small simple cycle separators for 2-connected planar graphs", *Journal of Computer and Systems Sciences* 32, 1986, pp. 265-279.

[OvL] Overmars, M.H., and van Leeuwen, J., "Maintenance of configurations in the plane", *Journal of Computer and Systems Sciences* 23, 1981, pp. 166-204.

[S] Snir, M., "On parallel searching", *SIAM Journal of Computing* 14, 3, August 1985, pp.688-708.

[TV] Tamassia, R. and Vitter, J.S., "Optimal parallel algorithms for transitive closure and point location in planar structures", Technical Report No. CS-89-20, Department of Computer Science, Brown University, March 1989.

[V] Venkatesan, S.M., "Improved constants for some separator theorems", *Journal of Algorithms* 8, 1987, pp. 572-578.

EUCLIDEAN SHORTEST PATH IN THE PRESENCE OF OBSTACLES

YONG-MAO CHEN AND PRAKASH RAMANAN *
Department of Computer Science
University of California
Santa Barbara, CA 93106

ABSTRACT

We consider the problem of finding the (Euclidean) shortest path $SP(s, t)$ between two points s and t in the plane, in the presence of obstacles. Lee and Preparata considered a special case, where the obstacles are n disjoint segments parallel to the y-axis. Let V be the set consisting of s, t and the endpoints of the n segments. They showed that the shortest path between s and any point in V is monotone with respect to the x-axis. Let $SPT(V)$ denote the Shortest Path Tree on V that consists of the shortest path from s to each point in V. They presented an $O(n \log n)$ algorithm for constructing $SPT(V)$, and an $\Omega(n \log n)$ lower bound for finding $SP(s, t)$. Their algorithm first sorts the segments according to their abscissae, and then constructs $SPT(V)$ by sweeping a vertical line from s to t. Their lower bound proof is based on the fact that any algorithm that finds $SP(s, t)$ must actually sort the segments according to their abscissae. We prove that even if the segments are given in sorted order of their abscissae, $\Omega(n \log n)$ time is still required to find $SPT(V)$. The sweep-line technique for finding $SP(s, t)$ can be used whenever $SP(s, t)$ is monotone. Lee and Preparata stated that $SP(s, t)$ is monotone for arbitrary obstacles, if the projections of any two obstacles on the x-axis do not overlap. We prove that $SP(s, t)$ is monotone if the projections of any two obstacles do not *partially* overlap, and s and t are respectively to the left and to the right of all the obstacles.

1. INTRODUCTION

In this paper, we consider the problem of finding the (Euclidean) shortest path $SP(s, t)$ between two points s and t in the plane, in the presence of obstacles. In the most widely studied version of the problem, the set of obstacles consists of n line segments which do not intersect each other, except possibly at their endpoints. For this case, let V be the set consisting of s, t and the endpoints of the segments. It is well-known that the shortest path between s and any point in V is a polygonal chain whose vertices are in V. Two points $p, q \in V$ are said to be *visible* to each other, if the line segment pq does not intersect the interior of any obstacle. Let the *Visibility Graph* be defined as $VG = (V, E)$, where the segment $pq \in E$ if and only if p and q are visible to each other. Most known algorithms for the problem (see [2,5]) first construct VG. Then, the shortest path between s and t can be found by applying Dijkstra's shortest path algorithm to this graph. In the worst case, these algorithms require $\Omega(n^2)$ time, since VG may have $\Omega(n^2)$ edges. Recently, Mitchell [4] presented an algorithm that does not explicitly construct VG, but his algorithm also requires $\Omega(n^2)$ time in the worst case.

---

*This author's research was supported in part by the National Science Foundation under grant CCR-88-03241.

A path P is said to be *monotone* with respect to a line L, if any line perpendicular to L cuts P at at most one point. Lee and Preparata [3] considered a special case of the shortest path problem, where the obstacles are disjoint segments parallel to the y-axis. They showed that the shortest path between s and any point in V is monotone with respect to the x-axis. Let $SPT(V)$ denote the *Shortest Path Tree* rooted at s, that consists of the shortest path from s to each vertex in V. They presented an $O(n \log n)$ algorithm for finding $SPT(V)$, and an $\Omega(n \log n)$ lower bound for finding $SP(s, t)$. Their algorithm first sorts the segments according to their abscissae, and then constructs $SPT(V)$ by sweeping a vertical line from s to t. Their lower bound proof is based on the fact that any algorithm that finds $SP(s, t)$ must actually sort the segments according to their abscissae.

In Section 2, we briefly describe Lee and Preparata's algorithm for constructing $SPT(V)$ when the obstacles are n vertical segments. In Section 3, we prove that even if the obstacle segments (parallel to the y-axis) are given in sorted order of their abscissae, $\Omega(n \log n)$ time is still required to find $SPT(V)$. Lee and Preparata stated that their sweep-line technique for finding $SP(s, t)$ can be used whenever $SP(s, t)$ is monotone; they stated that $SP(s, t)$ is monotone (w.r.t. the x-axis) for arbitrary obstacles, if the projections of any two obstacles on the x-axis do not overlap. We say that two segments *partially overlap* if they share an infinite number of points in common, but one segment is not contained in the other. In Section 4, we prove that $SP(s, t)$ is monotone if the projections of any two obstacles do not *partially* overlap, and s and t are respectively to the left and to the right of all the obstacles. Hence, Lee and Preparata's $O(n \log n)$ algorithm can be modified to find $SP(s, t)$, when the obstacles are n disjoint line segments, if the projections of any two obstacles do not *partially* overlap, and s and t are respectively to the left and to the right of all the obstacles; in fact, in $O(n \log n)$ time, we can construct the *Shortest Monotone Path Tree* (*SMPT*) rooted at s, that consists of the shortest monotone path from s to each point in V.

2. LEE AND PREPARATA'S ALGORITHM

In this section, we briefly explain Lee and Preparata's algorithm for finding $SPT(V)$ when the obstacles are n disjoint vertical segments. Without loss of generality, let s and t be to the left and to the right of all the obstacles, respectively. Lee and Preparata proved that the shortest path from s to any point in V is monotone with respect to the x axis. Hence, the shortest path tree from s to all the points in V with abscissa not exceeding $\overline{x}$ is not influenced by the segments lying to the right of $\overline{x}$. Hence, $SPT(V)$ can be constructed using a left-to-right line sweep from s to t.

Let $(l_1, l_2, \ldots, l_n)$ be the sequence of segments, in the order of increasing abscissae. For $1 \leq i \leq n$, let p_{2i-1} and p_{2i} be the lower and upper endpoints of l_i; let $p_0 = s$ and $p_{2n+1} = t$. Their algorithm constructs $SPT(V)$ from left-to-right, starting with s. Suppose that at some instant, we have grown the shortest path tree from s to all elements of V with abscissa not exceeding $\overline{x}$. Let $(l_1, l_2, \ldots, l_k)$ be the sequence of segments with abscissa not exceeding $\overline{x}$, and let $V_k = \{p_0, p_1, \ldots, p_{2k}\}$. Suppose that at this instant, we have a partition of the plane into regions R_i, $0 \leq i \leq 2k$, such that, in the absence of l_h, $k < h \leq n$, R_i is the locus of all the points in the plane for which the shortest path from s contains p_i as the second last vertex. Such a partition is called the *Shortest Path Map* (*SPM*(V_k)) of V_k. Now, we can extend the shortest path tree to l_{k+1} as follows: Locate p_{2k+1} and p_{2k+2} in $SPM(V_k)$; if p_{2k+1} lies in R_j and p_{2k+2} lies in R_m, then add the edges $p_j p_{2k+1}$ and $p_m p_{2k+2}$ to the shortest path tree.

We also need a mechanism to update the *SPM* to include the presence of l_{k+1}. First consider the *SPM* when there is only one obstacle; i.e., $V_1 = (s, p_1, p_2)$. $SPT(V_1)$ and $SPM(V_1)$ are shown in Figure 1. Dashed lines are used to represent the edges of $SPT(V_1)$, and solid lines are used to represent the edges of $SPM(V_1)$. Let $B(i, j)$ denote the boundary between regions R_i and R_j; $B(i, j)$ is called the ***bisector*** between regions R_i and R_j. With each p_i, associate a ***weight*** $w(p_i)$ which equals the length of the shortest path from s to p_i. Then $B(i, j)$ is the geometric locus $\{ q \,|\, w(p_i) + d(q, p_i) = w(p_j) + d(q, p_j)\}$. Since $d(q, p_i) - d(q, p_j) = w(p_j) - w(p_i)$ is a constant, $B(i, j)$ belongs to one of the two branches of the hyperbola with points p_i and p_j as foci, and eccentricity $d(p_i, p_j)/|w(p_i) - w(p_j)|$. Specifically, $B(i, j)$ belongs to the branch closer to p_i if $w(p_i) > w(p_j)$, and to the branch closer to p_j otherwise. Also note that if $w(p_i) = w(p_j)$, then $B(i, j)$ becomes the perpendicular bisector of the segment $p_i p_j$; if $d(p_i, p_j) = |w(p_i) - w(p_j)|$, then $B(i, j)$ degenerates to a half-line (eg. $B(0, 1)$ and $B(0, 2)$ in Figure 1).

Lee and Preparata proved that each bisector is monotone with respect to the x-axis, and for any point $z \in R_i$, the line segment zp_i lies entirely within R_i. They showed how to update the *SPM* and extend the *SPT* as the sweep-line is moved from s to t. Their entire algorithm runs in $O(n \log n)$ time.

3. LOWER BOUND FOR CONSTRUCTING THE SPT

In this section, we prove an $\Omega(n \log n)$ lower bound for constructing $SPT(V)$, in the ***decision tree model of computation*** [1], when the obstacles are n vertical segments given in the order of increasing abscissae. We will show that there are at least $n!$ different possible values for $SPT(V)$, and the lower bound will follow from this result.

We shall now present a procedure for constructing $n!$ different *SPT*s on n vertical segments chosen in the order of increasing abscissae. We only need to explain how to select the segments one after another, such that different selections lead to different *SPT*s. This is done recursively as follows. For $n = 1$, take an arbitrary vertical segment to the right of s. The *SPT* consists of edges (s, p_1) and (s, p_2). Assume that we have constructed $k!$ different *SPT*s on k segments. Starting with one such tree T_k, we will show how to select l_{k+1} in $k + 1$ different ways, each way leading to a different *SPT* on $k + 1$ segments. For each selection, p_{2k+1} and p_{2k+2} will be connected to some vertex v in $V_k^+ = \{ s, p_2, p_4, \ldots, p_{2k}\}$ in the resulting *SPT*; note that both p_{2k+1} and p_{2k+2} will be connected to the same vertex. Our procedure for selecting l_{k+1} is as follows.

```
choose a vertex $v \in V_k^+$
if $v = s$
    then (see Figure 2a) choose $l_{k+1}$ to the right of $l_k$ such that the edge $sp_{2k+1}$ is above
        all the edges of $T_k$
    else
        let $v = p_{2i}$ for some $i$, $1 \le i \le k$
        let $p_{2j}$ be the predecessor of $p_{2i-1}$ and $p_{2i}$ in $T_k$
        if no segment lies in the region bounded by $B(2j, 2i)$, $p_{2i}p_{2i-1}$ and $B(2j, 2i-1)$
            then (see Figure 2b)
                let $A_i$ be the asymptote to the curve $B(2i-1, 2i)$.
                through $p_{2i}$, construct a half-line $B_i$ to the right, parallel to $A_i$.
                through $p_{2i}$, construct a half-line $C_i$ to the right, that bisects the angle
        $B(2j, 2i)p_{2i}B_i$.
        choose $l_{k+1}$ to the right of $l_k$ such that $p_{2k+2}$ lies on $C_i$, and $p_{2k+1}$ lies on $B_i$.
    else (see Figure 2c)
        let $p_{2i}p_{2m}$ be the first edge of $T_k$ encountered by $B(2j, 2i)$, when $B(2j, 2i)$
        is rotated clockwise around $p_{2i}$.
        through $p_{2i}$, construct half-lines $D_{i(k+1)}$ and $E_{i(k+1)}$ to the right ($D_{i(k+1)}$
        above $E_{i(k+1)}$) that trisect the angle $B(2j, 2i)p_{2i}p_{2m}$.
        choose $l_{k+1}$ to the right of $l_k$ such that $p_{2k+2}$ lies on $D_{i(k+1)}$ and $p_{2k+1}$ lies on
        $E_{i(k+1)}$.
```

In order to prove the correctness of this procedure, we need the following facts.

Fact 1. Consider an *SPT* on one segment (see Figure 1). Neither $B(1, 2)$ nor its asymptote can intersect $B(0, 1)$ or intersect $B(0, 2)$.

Proof. Suppose that $B(1, 2)$ intersects $B(0, 1)$ at q. Then the paths sp_1q and sp_2q must have the same length. This is a contradiction, since sp_1q is a straight line segment while sp_2q is not. Hence $B(1, 2)$ cannot intersect $B(0, 1)$ or $B(0, 2)$, and the same holds for its asymptote. ■

Fact 2. Let p_{2j} and p_{2i} be as mentioned in the above procedure. Let R be the region to the right of p_{2i} that is bounded by $B(2j, 2i)$ and B_i. If $q \in R$, then the shortest path from s to q must pass through p_{2i}, independent of the presence of other obstacle segments in R.

Proof. By induction on i. Consider the case $i = 1$ (then $j = 0$; see Figure 1). If the shortest path from s to q does not pass through p_2, then it must intersect $B(0, 2)$ or B_1 at some point q'. It is easy to see that the shortest path from s to q' must pass through p_2.

For the general case, by the inductive hypothesis, the shortest path from s to q must pass through p_{2j}. The remaining proof is as in the previous paragraph, where p_{2j} plays the role of s. ■

Now, we are ready to prove the following Lemma.

Lemma 1. Using the above procedure recursively, we can construct $n!$ different *SPT*s on n vertical segments chosen in the order of increasing abscissae.

Proof. The proof is by induction on n. The base case $n = 1$ is easily seen to be true. Assume that the above procedure correctly constructs $k!$ different *SPT*s on k segments. Let T_k be one such tree. It is easily seen by Fact 1, and from the way T_k was constructed, that it is possible to choose l_{k+1} in $k + 1$ different ways as mentioned in the above procedure; i.e., the region in which the above procedure wants to place l_{k+1} can never be empty.

Let T_{k+1} be a tree obtained by the above procedure. From Fact 2, it follows that T_{k+1} is indeed a *SPT*. ■

Theorem 1. Any decision tree algorithm that computes the *SPT* of n vertical segments given in the order of increasing abscissae must have $\Omega(n \log n)$ height.

Proof. The decision tree must have at least as many leaves as there are solutions to the problem. Hence the result follows from Lemma 1. ■

Note that our proof of an $\Omega(n \log n)$ lower bound holds even if the vertical segments are further restricted in one (but NOT both) of the following two ways: (i) each segment is of unit length, (ii) the abscissa of l_i is i.

4. EXTENSION OF LEE AND PREPARATA'S TECHNIQUE

In this section, we consider arbitrary obstacles satisfying the following two conditions.

Assumption 1. The projections of any two obstacles on the x-axis do not partially overlap.

Assumption 2. s and t are respectively to the left and to the right of all the obstacles.

If either of these assumptions fails, then $SP(s, t)$ need not be monotone (w.r.t. the x-axis), even if the set of obstacles consists only of horizontal segments. We show that if both these assumptions hold, then $SP(s, t)$ is monotone. Let the *left* and *right endpoints* of an obstacle A, denoted by $lep(A)$ and $rep(A)$, be points on A with the smallest and largest abscissa, respectively.

Theorem 2. Under Assumptions 1 and 2, the shortest path from s to t is monotone with respect to the x-axis.

Proof. The proof is by contradiction. Clearly, two nonadjacent arc segments in $SP(s, t)$ can not have any point in common. For any two points q_1, q_2 on $SP(s, t)$, let $SP(q_1, q_2)$ be the portion of $SP(s, t)$ between q_1 and q_2. Suppose we traverse $SP(s, t)$ from s to t. Let v be the first point at which $SP(s, t)$ turns backward (see Figure 3). Also assume that in $SP(s, t)$, the arc segment e_1 immediately preceding v is above the arc segment e_2 immediately following v. There must exist an obstacle A that lies between e_1 and e_2, such that $x(rep(A)) = x(v)$; otherwise, we can obtain a shorter path by going vertically from an interior point in e_1 to an interior point in e_2. Among all such obstacles, let A be such that $x(lep(A))$ is the smallest; let $lep(A) = v'$. By Assumption 2, $x(v') > x(s)$. Also by Assumption 2, there exists a point $u \neq v$ on $SP(v, t)$ such that $x(u) = x(v)$; let u be the first such point when we traverse $SP(v, t)$ from v. Let q be the leftmost point on $SP(v, u)$.

Consider the case $x(v') \geq x(q)$. Let a and b be the first point on $SP(v, s)$ and $SP(v, q)$ respectively, at which the vertical line through v' intersects $SP(s, t)$. There must exist an obstacle B that cuts ab; otherwise, we can obtain a shorter path by going directly from a to b. The obstacles A and B violate Assumption 1.

So let $x(v') < x(q)$. There must exist an obstacle B that cuts vu; otherwise, we can obtain a shorter path by going directly from v to u. We have $x(lep(B)) \geq x(q) > x(v')$. The obstacles A and B violate Assumption 1. ■

Now, consider the case where the obstacles are disjoint line segments satisfying Assumptions 1 and 2. For this case, Lee and Preparata's $O(n \log n)$ algorithm can be modified to construct the *Shortest Monotone Path Tree* (*SMPT*) rooted at s that consists of the shortest monotone path from s to each point in V. From $SMPT(V)$, we can obtain $SP(s, t)$. The modifications required in the sweep-line process to handle this extension are straight forward. Also, when all the obstacle segments are vertical, the *SMPT* becomes the *SPT*. Hence, by Theorem 1, the above mentioned algorithm is optimal.

REFERENCES

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, Mass., 1974.
2. Takao Asano, Tetsuo Asano, L. Guibas, J. Hershberger and H. Imai, Visibility-Polygon Search and Euclidean Shortest Paths, *Proc*. 26th *Ann. IEEE Symp. on Foundations of Computer Science*, October 1985, pp. 155-164.
3. D. T. Lee and F. P. Preparata, Euclidean Shortest Paths in the Presence of Rectilinear Barriers, *Networks* 14 (1984), pp. 393-410.
4. J. Mitchell, New Algorithm for Shortest Paths, to appear.
5. E. Welzl, Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time, *Inform. Process. Lett.* 20 (1985), pp. 167-171.

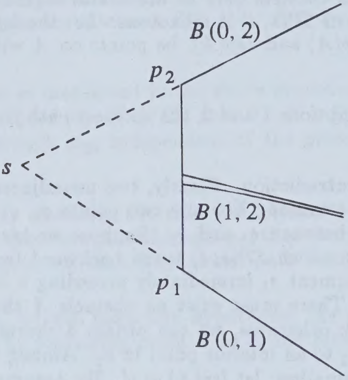


Figure 1

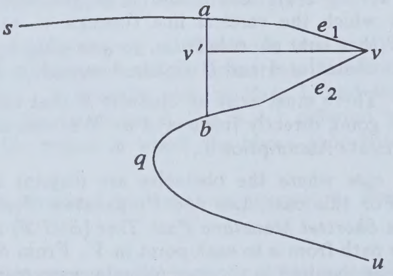
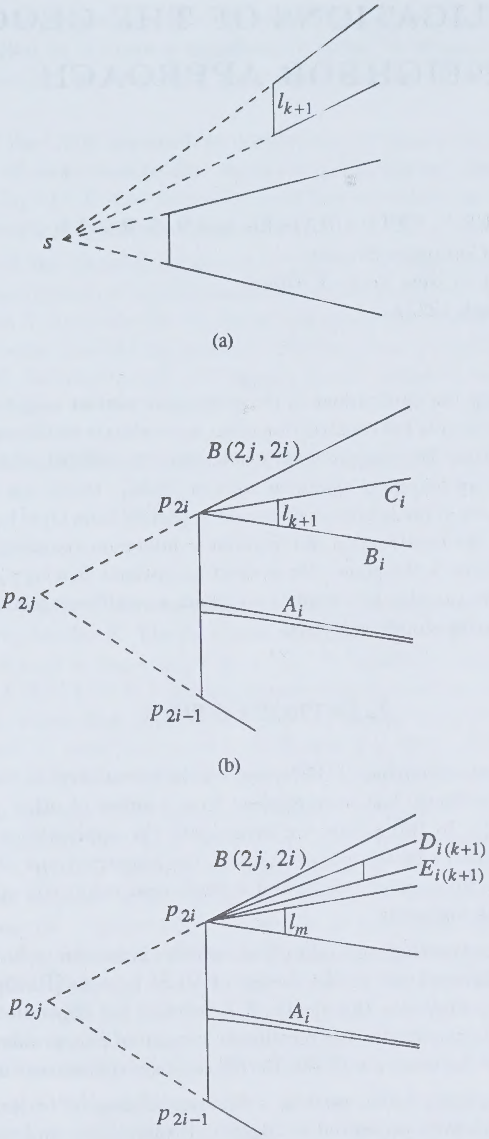


Figure 3


l_{k+1}
s
$B(2j, 2i)$
C_i
p_{2i}
l_{k+1}
B_i
p_{2j}
A_i
p_{2i-1}
$B(2j, 2i)$
$D_{i(k+1)}$
$E_{i(k+1)}$
p_{2i}
l_m
p_{2j}
A_i
p_{2i-1}

(a)

(b)

(c)

Figure 2

SOME APPLICATIONS OF THE GEOGRAPHIC NEAREST NEIGHBOR APPROACH

YOUNG C. WEE [1] , SETH CHAIKEN and S. S. RAVI [2]
Department of Computer Science
State University of New York at Albany
Albany, New York 12222

Abstract: We study the applications of the geographic nearest neighbor approach to two problems. The first problem is the construction of an approximate rectilinear Steiner tree for a set of n points in the plane. For this problem, we introduce a modified version of a data structure used for constructing minimum spanning trees in [Ya82]. Using this data structure, we improve the running times of the heuristics discussed in [Ber88] from $O(n^2 \log n)$ to $O(n \log^2 n)$. The second problem is the construction of a rectilinear minimum spanning tree for a set of n non-crossing line segments in the plane. We present an optimal $O(n \log n)$ algorithm for this problem. This algorithm can also be extended to obtain a rectilinear minimum spanning tree for a set of non-intersecting simple polygons.

1. INTRODUCTION

The *geographic nearest neighbor* (GNN) approach, introduced in [Ya82] for the minimum spanning tree problem, has been applied to a number of other problems (see for example, [Ka88, Cl87]). In this paper, we investigate the applications of this approach to two proximity problems in the plane, namely the constructions of an approximate rectilinear Steiner tree for a set of points and a rectilinear minimum spanning tree for a set of non-crossing line segments.

The problem of constructing minimum Steiner trees is known to be NP-hard [GJ79]. The applications of Steiner trees in the design of VLSI layouts [Ba83], communication networks [IW89], etc., motivate the study of heuristics for constructing approximate Steiner trees. Several heuristics for the rectilinear version of this problem have been proposed and studied (see for example [Ri89, Ber88] and the references contained therein).

Several greedy heuristics which work in a manner analogous to Kruskal's minimum spanning tree algorithm were presented in [Ber88]. Probabilistic and experimental analyses of these heuristics were carried out in [Ber88, BC86]. Although the performance of these heuristics was found to be good, their implementation requires $O(n^2 \log n)$ time. A heuristic due to Hannan was studied in [Ri89]. It was found that an efficient implementation of Hanan's heuristic runs faster than those studied in [Ber88, BC86] but its performance was worse. These observations raise the question of whether the greedy

---

[1] Supported in part by NSF Grants IRI-8703430 and CCR-8805782.
[2] Supported in part by NSF Grants DCI-8603318 and CCR-8905296.

heuristics could be implemented more efficiently. One of our results is that the GNN approach can be used to achieve a significantly faster implementation of the greedy heuristics. In particular, we show that these heuristics can be implemented to run in $O(n \log^2 n)$ time.

We also extend the GNN approach to compute a rectilinear minimum spanning tree (RMST) for a set of *non-crossing line segments* in the plane. Our algorithm runs in optimal time $O(n \log n)$. It can be easily modified to obtain an RMST for a set of non-intersecting simple polygons in the plane.

The remainder of this paper is organized as follows. Section 2 presents the necessary definitions, brief descriptions of the heuristics, data structures and the $O(n \log^2 n)$ implementations. Section 3 discusses the minimum spanning tree problem for line segments. Section 4 presents some concluding remarks. Due to space limitations, only sketches of proofs are included. Detailed proofs will appear in the complete version of the paper.

2. STEINER TREE HEURISTICS AND THEIR IMPLEMENTATION

2.1. *Preliminaries*

Throughout this paper, S denotes a set of n points in the plane. Let $MST(S)$ denote a *minimum spanning tree* for S. The L_1 distance (also called *rectilinear distance*) between two points p, q is $d(p,q) = |p_x - q_x| + |p_y - q_y|$. A *rectilinear minimum spanning tree* for S (denoted by $RMST(S)$) is a minimum spanning tree under rectilinear distance. A *Steiner tree* for S is a tree that spans S but may contain points not in S; such points are called *Steiner points.* A *rectilinear minimum Steiner tree* for S (denoted by $RST(S)$) is a Steiner tree whose length is minimum under rectilinear distance.

Our algorithms use the GNN approach [Ya82]. In order to discuss this approach, we need a few definitions. Let $\alpha(p)$ denote the closed region between the two rays in directions r_1 and r_2 from p forming an acute angle $\alpha = [r_1, r_2]$. A site q is said to be an *α-nearest neighbor* of p among the sites in S if and only if q is a nearest neighbor of p among $(S - \{p\}) \cap \alpha(p)$. Then the *all pairs geographic nearest neighbors problem* (GNN) is the following: Given an acute angular range α and a set of sites S, construct the set $\{(p,q) \mid q = \alpha\text{-nearest neighbor of } p \text{ in } S, \text{ for each } p \in S\}$. Let $\Delta_k = \{[2i\pi/k, 2(i+1)\pi/k] \mid 0 \le i < k\}$; that is, Δ_k is a k-equipartition of $[0, 2\pi]$. By $NN_k(p,S)$, we denote the set of edges from p to its α-nearest neighbor[3] in each $\alpha \in \Delta_k$. Then $NN_k(S)$ is a directed graph $G(V,E)$ where $V = S$ and $E = \bigcup_{p \in S} NN_k(p,S)$.

The directed graph NN_k and its modified versions to be introduced later satisfy several useful properties. In particular, our algorithms exploit the following two properties [Ya82, WCW88]:

1. $NN_k(S)$ has outdegree bounded by k.
2. Under any L_p metric, $NN_8(S)$ contains $MST(S)$.

---

[3] If there are two or more nearest neighbors in an angular region, we choose one arbitrarily.

The usefulness of these properties and a modified version of NN_k for the RST problem will be discussed in Section 2.3.

2.2. *Heuristics*

The heuristics studied in [Ber88, BC86] are based on Kruskal's MST algorithm. Recall that Kruskal's algorithm repeatedly adds an edge of least cost connecting two different components until only one connected component is left. In the construction of Steiner trees for a set S, a connected component contains either a single point of S or two or more points of S connected together by *edges*. An edge under the L_1 metric is L-shaped; it consists of a horizontal line segment and a vertical line segment (except in degenerate cases). The point at which the two line segments intersect is called a *corner point* which will be a Steiner point. A *vertex* is either a point in S or a Steiner point. The set of Steiner points created by the construction depends on the order in which the components are joined.

We now discuss briefly the heuristics studied in [Ber88, BC86]. For more details, we refer the reader to these references.

Heuristic A: The first edge joins two closest points in S. Each subsequent step joins two closest components by an edge. This edge may connect a vertex to a vertex, a vertex to a line or a line to a line. These connections may create additional Steiner points.

Heuristic B: This is similar to Heuristic A except that an edge can only be a vertex to vertex connection.

In [Ber88] a third heuristic (called Heuristic C) was introduced for purposes of simplifying the probabilistic analysis. It was shown that the performances of these heuristics are similar and that there are no dominance relations among them. The implementations of these heuristics presented in [Ber88, BC86] take $O(n^2 \log n)$ time [Ri89]. In this paper, we show that the Heuristics A and B can be implemented in time $O(n \log^2 n)$. For brevity, we will present the implementation of Heuristic B. With some modifications of this algorithm, it is possible to implement Heuristic A with the same running time.

2.3. *Data Structures*

The *relative neighborhood graph* of S denoted by $RNG(S)$ [To80] is defined as follows: for $p, q \in S$, $\overline{pq} \in RNG(S)$ if and only if for all $r \in S - \{p, q\}$, $d(p,q) \leq d(p,r)$ or $d(p,q) \leq d(q,r)$. It is well known that $MST(S) \subseteq RNG(S)$ under *any* L_p metric [To80, Su83, JK87]. This containment has also been shown for *any* symmetric distance function d [WCW88].

Recall that the heuristics described in Section 2.2 are based on Kruskal's algorithm for constructing minimum spanning trees. We obtain faster implementations of these heuristics by dynamically maintaining a heap with only a *linear* number of edges. In the remainder of this section, we will discuss why a linear number of edges suffice. We begin with a property satisfied by the NN_k data structure mentioned above.

Lemma 2.1 *For any L_p metric,*

$$MST(S) \subseteq RNG(S) \subseteq NN_8(S)$$

Proof. Similar to that of Theorem 2.2 below. □

Each node of the directed graph $NN_k(S)$ has outdegree bounded by k (a constant). As a result, the number of edges in the directed graph $NN_k(S)$ is $O(n)$. Thus NN_k is a suitable data structure for the *static version*[4] of the RST problem. However, the indegree of a node in NN_k can be $\Omega(n)$. As a consequence, NN_k cannot be directly used for the *dynamic version* since one insertion may cause $\Omega(n)$ new edges. For the same reason, an approach based on the *Voronoi diagram* is also not suitable for the dynamic version. Our approach uses a modified version of NN_k, called DNN_k, defined as follows:

1. $S^{i+1} = S^i \cup \{r\}$
2. $DNN_k(S^{i+1} \cup \{r\}) = DNN_k(S^i) \cup NN_k(r, S^i)$

Initially when S is empty, $DNN_k(S)$ is also empty. Thus, whenever a given point or new Steiner point r is added, only a constant number (k) of outgoing edges from r are added into DNN_k. As a result, $DNN_k(S')$ will contain only $O(n)$ edges as long as the number of points inserted is $O(n)$. This condition will obviously hold, because the number of Steiner points is at most the number of edges in the final tree. Now we show that DNN_8 also satisfies a property similar to NN_8 (Lemma 2.1).

Theorem 2.2 *For any L_p metric,*

$$MST(S) \subseteq RNG(S) \subseteq DNN_8(S)$$

Proof. Induction on the size of S. The base case for $|S| = 2$ is trivial. Assume that $RNG(S) \subseteq DNN_8(S)$. Thus, for any $\overline{pq} \in RNG(S)$, either $\overrightarrow{pq}$ or $\overrightarrow{qp}$ is in $DNN_8(S)$.

Consider $S' = S \cup \{r\}$, $r \notin S$. To show $RNG(S') \subseteq DNN_8(S')$, it suffices to show that if $\overline{rp}$ in $RNG(S')$, then $\overrightarrow{rp}$ is in $DNN_8(S')$. Suppose $\overline{rp} \in RNG(S')$ and $\overrightarrow{rp} \notin DNN_8(S')$. Then there must exist $t \in S$ such that $d(r,t) < d(r,p)$ and $\angle trp \leq \pi/4$. It is easy to verify that $d(t,p) < d(r,p)$. Therefore, $\overline{rp}$ is not in $RNG(S')$, which is a contradiction. □

Theorem 2.2 shows that in each stage of the implementation of Heuristic B, it is enough to maintain in a heap the edges of $DNN_8(S')$ which contain $MST(S')$ where $S' = S \cup \{\text{Steiner points added so far}\}$.

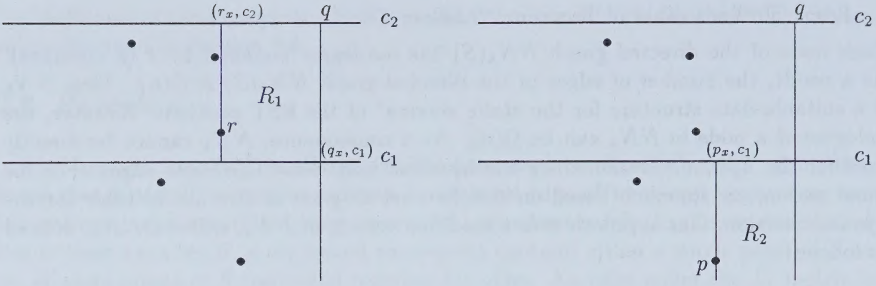
2.4. *An Efficient Implementation*

It is known that Kruskal's algorithm can be implemented in $O(e \log^* n)$ time [FT87] where e represents the number of edges in the graph. Thus Theorem 2.2 in conjunction with the fact that $DNN_8(S')$ has only $O(n)$ edges allows us to conclude that the running time of heuristic B is $O(n \log^* n + f(n))$ where $f(n)$ represents the time needed to compute $DNN_8(S')$. In this section, we show how $DNN_8(S')$ can be computed efficiently.

First, we show that under L_1 metric, the GNN problem (or equivalently, the NN_k computation) can be reduced to the *dominance counting problem* (to be discussed below).

---

[4] The Steiner points are given and no new Steiner points are added.



(a) (b)

Figure 1.

The *range tree* used for the dominance counting problem can be constructed in $O(n \log n)$ time so that a query can be answered in $O(\log n)$ time [Wi85]. Then, using *static-to-dynamic transformation* [Be79] (for additional references see [EO85, WW88]), an $O(n \log^2 n)$ algorithm for computing $DNN_8(S')$ is obtained. To simplify the presentation of our algorithm, we will assume the angular range $\alpha = [\pi, 3\pi/2]$. The algorithm can be easily modified to work for any α where $|\alpha| \leq \pi/2$ by suitable rotations of the x and y axes.

A point p *dominates* q if and only if $p_x \geq q_x$ and $p_y \geq q_y$. Given a set S of points, the *dominance counting problem* asks for $Count(q, S)$, the number of points in S dominated by a query point q. We assume that the reader is familiar with the data structures (range trees) and the algorithms for the dominance counting problem discussed in [Be80, Wi85] and so only focus on the necessary invariants for the operator "+". We discuss these invariants in some detail because they are used (in a slightly modified form) in Section 3 for the computation of an MST for a *set of non-crossing lines.*

Invariants. Let c_1 and c_2 be two real numbers such that $c_1 \leq c_2$. Further, let $S = S_1 \cup S_2$ where $S_1 = S \cap \{y \leq c_1\}$ and $S_2 = S \cap \{c_1 < y \leq c_2\}$ (See Figure 1). Given any point q on the line $y = c_2$, let p and r be the immediate x-predecessors of q in S_1 and S_2 respectively. Then the invariants of "+" opertor for the dominance counting problem are:

$$Count(q, S) = Count((q_x, c_1), S_1) + Count((r_x, c_2), S_2) \tag{1}$$

and

$$Count(q, S) = Count((p_x, c_1), S_1) + Count(q, S_2) \tag{2}$$

Equations 1 and 2 correspond to Figure 1a and Figure 1b respectively. The correctness of these equations is due to the following fact: the rectangular regions $R_1 = \{(x, y) \mid r_x < x < q_x, c_1 < y < c_2\}$ and $R_2 = \{(x, y) \mid p_x < x < q_x, y < c_1\}$ do not contain any point of S.

Let $Near(p, S)$ denote the α-nearest neighbor of p among the points in S. To reduce the GNN problem to the dominance counting problem, it is sufficient to show that the above equations hold with $Count$ replaced by $Near$ and an appropriate definition of "+". Under the L_1 metric, it is possible to define a binary operation "+" that can be carried out in $O(1)$ time. In particular, the sum "$Near((q_x, c_1), S_1) + Near((r_x, c_2), S_2)$" can be found by the following $O(1)$ operations. Calculate the distance from q to the two points $Near((q_x, c_1), S_1)$ and $Near((r_x, c_2), S_2)$ and set $Near(q, S)$ to be the closer of these two points. The correctness of the above operations follows from the fact that under the L_1 metric, if a is dominated by b and b is dominated by c then $d(a, c) = d(a, b) + d(b, c)$ [Cl84]. Similarly, we can carry out the computation in Equation 2 in $O(1)$ time. Since the computation of $NN_k(S)$ can be accomplished by solving k instances of the GNN problem, we see that the NN_k computation is reduced to the dominance counting problem.

The "+" operator defined above for the NN_k problem is clearly associative and commutative. Therefore the static-to-dynamic transformation of [Be79] can be applied to NN_k. Thus query and insertion operations for a Steiner point into $DNN_k(S')$ can be done in $O(\log^2 n)$ time. Since the number of Steiner points inserted is less than n, it follows that the time spent in the computation of DNN_k is $O(n \log^2 n)$. By Theorem 2.2 and the above discussion, we conclude that heuristic B can be implemented in $O(n \log^2 n)$ time.

Recall that in the implementation of Heuristic B, we made use of the fact that for each insertion of a Steiner point, we need add only a constant number of edges into the heap. In order to implement Heuristic A, we need a similar fact except that instead of a Steiner point, a line gets inserted. This fact is provided by the following lemma.

Lemma 2.3 *For any line segment l in the RST constructed by Heuristic A, at most 4 vertices are connected to the interior*[5] *of l.*

Proof. Omitted. The idea is to divide the region around the line segment l into subregions and then bound the number of points that can be connected to the interior of l in each subregion using the fact that Kruskal's algorithm adds edges in non-decreasing order of length. □

With the above lemma, the modifications needed for implementing Heuristic A are similar to those of Heuristic B. The details will be included in the complete version of this paper.

3. MST FOR A SET OF NON-CROSSING LINE SEGMENTS

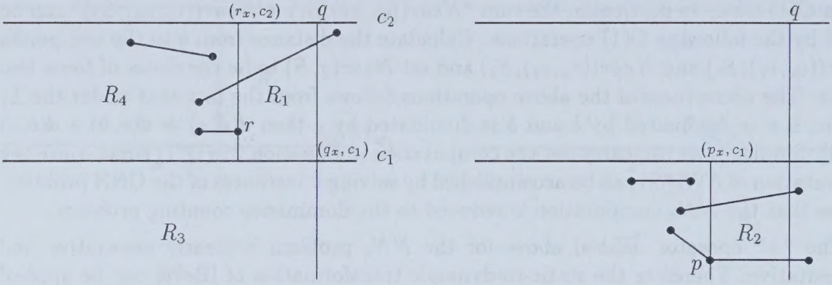
In this section, we extend RNG and NN_k concepts for computing an MST of a set T of non-crossing line segments. Note that edges in $MST(T)$ may connect a vertex to a vertex, a vertex to a line or a line to a line. We then reduce the $NN_k(T)$ computation to the dominance counting problem with some modifications of the range tree and obtain an $O(n \log n)$ time algorithm for computing $NN_k(T)$ under L_1 metric (and thus $RMST(T)$).

Fact 3.4 *Under any L_p metric, a minimum distance connection between two line segments can always be chosen to be an endpoint-to-line*[6] *connection.*

---

[5] The *interior* of a line segment l is the set of points in l except for its endpoints.

[6] Here a line is assumed to include its endpoints.



(a) (b)

Figure 2.

Proof. (*Sketch*) Consider an edge connecting two lines. It can be slid to touch an endpoint without increasing its length. □

Thus there always exists a minimum spanning tree which has only endpoint-to-line connections. This observation enables us to extend the RNG and NN_k concepts to a set of line segments. Define NN_k of a set T of line segments so that each endpoint in T has edges to its k geographic nearest lines.

Theorem 3.5 *For any L_p metric,*

$$MST(T) \subseteq RNG(T) \subseteq NN_8(T)$$

Proof. Follows from Fact 3.1 and Lemma 2.1. □

The algorithm presented in Section 2.4 for computing NN_k cannot be directly applied to a set of line segments because the regions R_1 and R_2 in Figure 1 may not be empty when T is a set of line segments (i.e., line segments may straddle R_1 or R_2 – see Figure 2). Thus the invariants in Section 2 need to be modified for the GNN computation on a set of lines.

Given a point p, let $Near(p, T)$ denote the line segment nearest to p among those in T. Consider the situation in Figure 2a where the lines straddle the region R_1. There cannot be any line segment whose endpoint lies inside the region R_1 since the endpoint r is the immediate x-predecessor of q in $\{$ endpoints of T $\} \cap \{(x, y) \mid c_1 < y \le c_2\}$. Furthermore, the lines in T are non-crossing. Thus, $Near((r_x, c_2), T \cap R_1) = Near(q, T \cap R_1)$. Let $R_3 = \{(x, y) \mid x \le q_x, y \le c_1\}$ and $R_4 = \{(x, y) \mid x \le r_x, c_1 < y \le c_2\}$. Then the following equation, where the operator "+" is interpreted in a manner similar to that in Section 2.4, holds:

$$Near(q, T) = Near((q_x, c_1), T \cap R_3) + Near((r_x, c_2), T \cap R_4) + Near((r_x, c_2), T \cap R_1) \quad (3)$$

A similar equation holds for the situation in Figure 2b where the lines straddle the region R_2. Thus, we store in each node v of the auxiliary fields in the range tree, the nearest line segment among those straddling the right boundary of $\alpha(v)$ in addition to the nearest among those in $\alpha(v)$.

Theorem 3.6 *Under L_1 metric, NN_k of n non-crossing line segments can be computed in $O(n \log n)$ time and the nearest line query can be answered in $O(\log n)$ time.*

Proof. Follows from the above discussion. □

4. SUMMARY

In this paper, we showed that the geographic nearest neighbor approach is useful in the design of efficient algorithms for the construction of approximate rectilinear Steiner trees. We also showed that the approach can be used to compute an RMST for a set of non-crossing line segments in $O(n \log n)$ time. It is possible to extend the latter algorithm to compute an RMST for a set of non-intersecting simple polygons. Our algorithms can also be extended to higher dimensions.

References

[Ba83] A. Baratz, Algorithms for Integrated Circuit Signal Routing, Ph.D. Thesis MIT/LCS/TR-248, October 1983.

[BC86] M. W. Bern and M. de Carvalho, A Greedy Heuristic for the Rectilinear Steiner Tree Problem, Technical Report No. UCB-CSD-87-306, Computer Science Division, Univ. of California, Berkeley, California, 1986.

[Be79] J. L. Bentley, Decomposable Searching Problems, *Info. Proc. Lett.* **8**, 244-251, 1979.

[Be80] J. L. Bentley, Multidimensional Divide and Conquer, *CACM* **23(4)**, 214-229, 1980.

[Ber88] M. W. Bern, Two Probabilistic Results on Rectilinear Steiner Trees, *Algorithmica* **3**, 191-204, 1988.

[Cl84] K. Clarkson, Fast Expected-time and Approximation Algorithms for Geometric Minimum Spanning Trees, *Proc. 16th Ann. ACM Symp. Theory of Computing*, 342-348, 1984.

[Cl87] K. Clarkson, Approximation Problems for Shortest Path Motion Planning, *Proc. 19th Ann. ACM Symp. Theory of Computing*, 56-65, 1987.

[EO85] H. Edelsbrunner and M. H. Overmars, Batched Dynamic Solutions to Decomposable Searching Problems, *J. Algorithms* **6**, 515-542, 1985.

[FT87] M. L. Fredman and R. E. Tarjan, Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms, *JACM*, **34**(3), 596-615, 1987.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, San Fransisco, 1979.

[IW89] M. Imase and B. M. Waxman, Dynamic Steiner Tree Problem, Technical Report No. WUCS-89-11, Dept. of Computer Science, Washington Univ. at St. Louis, MO, March 1989.

[JK87] J. W. Jaromczyk and M. Kowaluk, A Note on Relative Neighborhood Graphs, *Proc. Third Ann. Symp. on Computational Geometry*, 233-241, 1987.

[Ka88] J. Katajainen, The Region Approach for Computing Relative Neighborhood Graphs in the L_p Metric, *Computing* **40**, 147-161, 1988.

[Ri89] D. Richards, Fast Heuristic Algorithms for Rectilinear Steiner Trees, *Algorithmica* 4, 191-207, 1989.

[Su83] K. J. Supowit, The Relative Neighborhood Graph with an Application to Minimum Spanning Trees, *J. ACM 30*(**3**), 428-447, 1983.

[To80] G. T. Toussaint, The Relative Neighborhood Graph of a Finite Planar Set, *Pattern Recognition* **12**, 261-268, 1980.

[WCW88] Y. C. Wee, S. Chaiken and D. E. Willard, The Angle Restricted Voronoi Diagram, *First Canadian Conference on Computational Geometry*, Montreal, 1989, Technical Report No. 88-31 State Univ. of N.Y. at Albany.

[Wi85] D. E. Willard, New Data Structure for Orthogonal Range Queries, *SIAM Journal on Computing* **14**, 233-253, 1985.

[WW88] D. E. Willard and Y. C. Wee, Quasi-Valid Range Querying and its Implications for Nearest-Neighbor Problems, *Proc. 4th Annual ACM Symp. on Computational Geometry*, 34-43, 1988.

[Ya82] A. C. Yao, On Constructing Minimum Spanning Trees in k-dimensional Spaces and Related Problems, *SIAM J. Comput.* **11**(4), 721-736, 1982.

OUT-OF-ROUNDNESS PROBLEM REVISITED

V. B. LE
AT&T Bell Laboratories
2000 N. Naperville Road,
Naperville, IL., 60566-7033

D. T. LEE †
Department of Electrical Engineering and Computer Science
The Technical Institute
Northwestern University
Evanston, IL., 60208

ABSTRACT

The Out-Of-Roundness measurement of a circular profile undertakes different schemes depending on the type of center specified. The most common standard recommended by the American National Standards Institute (ANSI) is the Minimum Radial Separation center. In this paper, we introduce another standard called the Minimum Area Difference center. Although the two centers are different in characteristics, the approach to finding both centers shares many commonalities. We present an O(nlogn + k) time algorithm to compute the Minimum Radial Separation center, and the Minimum Area Difference center of a simple polygon G, where n is the number of vertices of G, and k is the number of intersection points of the Medial axis (or skeleton region elements) and the farthest neighbor Voronoi diagram of G.

1. INTRODUCTION

In the process of mechanical design, the dimensioning of components and assemblies play a major role to retain control of a design. One of many pertinent factors of dimensioning is the tolerance factor. Tolerance, in this context, is the amount of variation in a dimension which can be permitted, or tolerated without impairing the functional fitness of the component. To measure if a component satisfies certain specified tolerance factor requires a precise and efficient measuring tool and metric scheme.

In this paper, we study the scheme to measure the tolerance factor of a circular profile also known as the Out-of-Roundness factor. Depending on the type of center specified, the Out-of-Roundness factor of the same circular profile may have different values. Four types of centers are mostly used in mechanical design - the Maximum Inscribed Center (MIC), the Minimum Circumscribed Center (MCC), the Least Square Center (LSC), and the Minimum Radial Separation center (MRS) [6]. The MIC center defines the largest circle inscribed within the measured polar profile. By contrast, the MCC circle is the smallest circle circumscribed just the measured profile. The LSC center specifies a circle from which the sum of the squares of the radial ordinates of the measured polar profile has a minimum value. For the MRS case, the radial difference between concentric circles which just contain the measured polar profile is a minimum. As recommended by the American National Standards Institute (ANSI Standard B89.3.1-1972), the default standard to measure Out-of-Roundness is the Minimum Radial Separation center.

The MRS center of a simple polygon can be determined by using a rather primitive method. The contour of the polygon is first plotted on a polar chart. The MRS center is then manually located by using concentric circles etched on a mylar overlay placed on top of the chart. The Out-of-Roundness

---

† Supported in part by the National Science Foundation under the Grant No. DCR-8420814.

can be inferred from the difference between the radii of the two concentric circles. It is obvious that this method is neither accurate nor efficient. Attempts have been made to solve the location of the MRS center by relying on computer programs adopting the Minimum Square Root method. The X and Y coordinates of the center are determined by respectively averaging the X and Y coordinates of the points around the contour. However, such an approach only gives a coarse approximation of the center and by no means satisfies the definition of the MRS center. The problem of determining the exact MRS center remains open. Recently, Ebara et al [3] have solved the roundness problem under the context of a point set. But the algorithm of point set is not viable for the case of simple polygon, since it may lead to an unacceptable result. This discrepancy will be discussed in section 4.

The results described in sections 3 and 4 show the MRS problem is solvable under the framework of computational geometry. In fact, the MIC and MCC centers can be solved respectively in O(nlogn) (see [10] for example) and O(n) [2,8] time using techniques developed in computational geometry. It can be proven that there exists an MRS center on the medial axis of the polygon. There are certain configurations where such an MRS center may not be unique. In this paper, the computation of the MRS center is based on the medial axis and the farthest neighbor Voronoi diagram of the polygon. The complexity of the algorithm can be proven to be solvable in O(nlogn + k) time, where n is the total number of vertices in the polygon, and k is the number of intersections between the medial axis and the farthest neighbor Voronoi diagram, which will be defined in the next section.

By the same token, we define a different type of center called the Minimum Area Difference (MAD) center. The MAD center of a simple polygon has a similar characteristic, and the finding of the center makes use of the information of the skeleton region elements and the farthest neighbor Voronoi diagrams of the polygon. The topic of MAD center is discussed in section 5.

2. DEFINITIONS AND NOTATIONS

DEFINITION 1: An inscribed circle $IC_G(C,R)$ of a simple polygon G is a circle inscribed in G and is defined by a center C in G and a radius R.

DEFINITION 2: A circumscribed circle $CC_G(C,R)$ of a simple polygon G is a circle circumscribing G and is defined by a center C in G and a radius R.

The subscript G for polygon G will be omitted when it is understood.

DEFINITION 3: The Minimum Radial Separation center of a simple polygon G, denoted by MRS, is defined as the center of two concentric circles, one being the inscribed circle $IC(C,R_1)$ and the other being the circumscribed circle $CC(C,R_2)$, such that the radial separation of the two concentric circles is minimum. The difference of the two radius $(R_2 - R_1)$ is called the MRS value of the polygon.

In fact, the MRS center is equivalent to the 1-Circle approximation center satisfying the following constraint:

$$\underset{c_j \in R(P)}{Min} \quad \underset{p_i \in B(P)}{Max} \; d(P_i, C(c_j, r))$$

where R(P) and B(P) respectively are the interior region and the boundary defined by the polygon P. The function d(p,C) denotes the euclidean distance of the point p to the circle C.

DEFINITION 4: The Minimum Area Difference center of a simple polygon G, denoted by MAD, is defined as the center of two concentric circles, one being the inscribed circle $IC(C,R_1)$ and the other being the circumscribed circle $CC(C,R_2)$, such that the area difference of the two concentric circles is minimum. The area difference, $\Pi(R_2^2 - R_1^2)$, is called the MAD value of the polygon.

DEFINITION 5: The medial axis (also known as skeleton) of a simple polygon G, denoted by MA(G), is the set of points {q} internal to G such that there are at least two points on the boundary of the polygon that are equidistant from {q} and are closest to {q}. MA(G) partitions the interior of the polygon into regions called skeleton regions (SRs). Each skeleton region SR_i is bounded by boundary edges and medial axis segments.

Each skeleton region is composed of one or many skeleton region elements SREs depending on the number of boundary edges and *reflex*[1] vertices on the boundary of the skeleton region. Each skeleton region element will contain either one boundary edge or one reflex vertex.

DEFINITION 6: Given two points p and q, the half-plane associated with p denoted by h(p,q) is the locus of points closer to p than q, and is defined by h(p,q) = {r | d(r,p) ≤ d(r,q)}, where d(p,q) denotes the euclidean distance between points p and q.

DEFINITION 7: Given a set of points S = $\{p_1, p_2, \ldots, p_n\}$, the farthest neighbor Voronoi polygon associated with a point p_i, denoted by FNV(p_i), is the locus of points farthest from p_i than any other point in S and is defined as below:

$$FNV(q_i) = \bigcap_{j \neq i} h(q_j, q_i)$$

Note that FNV(p_i) may be empty.

DEFINITION 8: Given a set of points S = $\{p_1, p_2, \ldots, p_n\}$, the farthest neighbor Voronoi diagram of S, denoted by FNVOD(S), is a collection of non-empty FNV polygons and is defined as below:

$$FNVOD(S) = \bigcup_{i=1}^{n} \left\{ FNV(q_i) \mid FNV(q_i) \neq \emptyset \right\}$$

The edge B(p_i,p_j) shared by two FNV polygons FNV(p_i) and FNV(p_j) is called the farthest neighbor Voronoi edge or FNV edge; the *points*[2] shared by three FNV polygons are called the farthest neighbor Voronoi points or FNV points.

DEFINITION 9: Given a set S = $\{p_1, p_2, \ldots, p_n\}$, the convex hull CH(S) of S is the smallest convex set containing S. The edges and points of the convex hull are called hull edges and hull points respectively.

It can be shown [4,9] that FNVOD(S) consists of only polygons FNV(p_i) whose associated points are on the convex hull CH(S) defined by S.

3. PROPERTIES OF MRS CENTER

Unless explicitly specified, we use the term "vertex" to indicate a vertex of a polygon, and the term "perpendicular to the boundary" to mean either perpendicular to an edge of the boundary of the polygon or passing through a reflex vertex of the polygon.

Lemma 1 states the relationship of an MRS center to one of its specific vertices on the circumscribed circle. The specific vertex is essential to determining the validity of the locus of certain MRS center.

LEMMA 1: If an MRS center C_{mrs} lies inside a skeleton region SR_i, then there exists a vertex p_* on CC(C_{mrs},R) such that the segment $p_* C_{mrs}$ is perpendicular to the boundary of SR_i. The vertex p_* is called the **critical vertex** of C_{mrs}.

PROOF: Let C_{mrs} be the MRS center in SR_i with circumscribed circle CC(C_{mrs},R_2) and inscribed circle IC(C_{mrs},R_1). Let S be the set of vertices of the simple polygon G, and $S_1 \subseteq$ S be the set of vertices on CC(C_{mrs},R_2). Suppose that none of the points $p_i \in S_1$ is the critical vertex p_* of C_{mrs}. Let q_1 and q_2 be the intersections of the line $p_* C_{mrs}$ with the medial axis and the boundary edge of SR_i respectively (Figure 3.1). For any vertex $p_i \in S_1$, the following triangle inequality holds:

$$q_1 p_i < C_{mrs} p_i + q_1 C_{mrs}$$

Replace $q_1 C_{mrs}$ by $q_1 q_2 - C_{mrs} q_2$, and rearrange the inequality:

---

(1) A vertex of a polygon is reflex if its internal angle is greater than Π .

(2) It is assumed that no more than three points of S are cocircular.

$$q_1p_i - q_1q_2 < C_{mrs}p_i - C_{mrs}q_2 \equiv R_2 - R_1 \quad (1)$$

The same argument applies to any vertex $p'_j \in S - S_1$. Indeed, let p'' be the intersection of C_{mrs}p' and the circumscribed center CC(C_{mrs},R_2). Inequality (1) still holds with a minor modification to the relation (from $<$ to $\leq$). This modification is necessary since p'' can be the critical vertex p_*.

$$q_1p'' - q_1q_2 \leq R_2 - R_1 \quad (2)$$

Since the length of q_1p'' is greater than q_1p', the following inequality holds:

$$q_1p' - q_1q_2 < q_1p'' - q_1q_2 \quad (3)$$

Inequalities (2) and (3) give:

$$q_1p' - q_1q_2 < R_2 - R_1 \quad (4)$$

Inequalities (1) and (4) imply that the radial separation value induced by q_1 is smaller than the MRS value of C_{mrs}. This contradicts the assumption that C_{mrs} is the MRS center. □

Based on the result of lemma 1, we can form the following theorem. Let S_1 be the set of vertices on the circumscribed circle CC(C_{mrs},R). Let q_1, and q_2 be the intersections of the line p_*C_{mrs} with the medial axis and the boundary edge of SR_i respectively. Let q_3 be the intersection, if any, of the farthest neighbor Voronoi edge of G with the segment $C_{mrs}q_2$ (Figure 3.2).

THEOREM 1: If C_{mrs} lies inside a skeleton region SR_i, then the following propositions hold:

1. If $|S_1| \geq 2$: L(C_{mrs}) = q_1C_{mrs}.
2. If $|S_1| = 1$: L(C_{mrs}) = q_1q_3 if $q_3 \neq \emptyset$, and L(C_{mrs}) = q_1q_2 otherwise

where L(C_{mrs}) is the locus of the MRS center, i.e., any point in L(C_{mrs}) can be used as an MRS center.

PROOF: (Case 1: $|S_1| \geq 2$) According to lemma 1, there exists a vertex in S_1 which is the critical vertex p_*. It is obvious that any point $t_i \in q_1C_{mrs}$ will have the same radial separation value, since the circumscribed radius and the inscribed radius increase or decrease by the same amount. The locus cannot extend beyond segment q_1C_{mrs}. Indeed, any point t' exiting q_1 will have a smaller inscribed radius (closer to the boundary edge) and a larger circumscribed radius (farther from p_*). Similarly, any point t'' exiting C_{mrs} will have a smaller inscribed radius (closer to the boundary edge) and a larger circumscribed radius (farther from a vertex $\in S_1 - \{p_*\} \neq \emptyset$).

(Case 2: $|S_1| = 1$) As in case 1, L(C_{mrs}) contains the segment q_1C_{mrs}, but L(C_{mrs}) is not constrained by the end point C_{mrs}. It can expand beyond C_{mrs} till whichever q_3 or q_2 is encountered first. If the FNV edge does intersect the segment $C_{mrs}q_2$, then any point t''' $\in q_3q_2$ will have a smaller inscribed radius and a larger circumscribed radius. On the other hand, if the point q_3 lies outside SR_i, then the whole segment $C_{mrs}q_2$ will have the same radial separation value as C_{mrs}. □

Theorem 1 implies that we can always find an MRS center on the medial axis determined by the intersection of the segment p_*C_{mrs} and the medial axis.

COROLLARY 1: If L(C_{mrs}) = $\{C_{mrs}\}$, then the MRS center C_{mrs} has to lie on the medial axis of the simple polygon G.

Based on the result of theorem 1 and corollary 1, we have the following:

THEOREM 2: There always exists a point on the medial axis MA(G) which defines the MRS center of the simple polygon G.

Theorems 1 and 2 indicate that, in general, the MRS center of a polygon may not be unique. Example 3.1 illustrates a case of non-unique MRS center where both C_1 and C_2 can be the MRS centers of the simple polygon.

For the case of convex polygons, where all vertices of the polygon are convex, the following Lemma always holds:

LEMMA 2: The MRS center of a convex polygon G has to lie on the medial axis and nowhere else.

PROOF: Since no vertex of the polygon is reflex or can be a critical vertex, L(C_{mrs}) = $\{C_{mrs}\}$ must be on the medial axis. □

Furthermore we have the following theorem:

THEOREM 3: The MRS center of a convex polygon G is at an intersection point of the medial axis MA(G) and the farthest neighbor Voronoi diagram FNVOD(G) of the polygon.

PROOF: Based on Lemma 2, the MRS center C_{mrs} is located on the medial segment S_i of a skeleton region SR(j). Suppose C_{mrs} lies inside FNV(p_i). Since G is a convex polygon, the segment $C_{mrs}p_i$ is entirely contained inside G (Figure 3.3). By choosing a point C_1 on C_{mrs}q which lies inside the intersection of FNV(p_i) and SR(j). The radial separation value MRS_{mrs} of C_{mrs} is determined by $C_{mrs}p_i - R_1$, and the radial separation value RS_1 of C_1 is $C_1p_i - R_2$, where R_1 and R_2 are the radii of the inscribed circles of C_{mrs} and C_1 respectively. The difference of the two radial separation values is given by:

$$MRS_{mrs} - RS_1 = C_{mrs}p_i - R_1 - C_1p_i + R_2 = C_{mrs}C_1 - R_1 + R_2$$

Since p_i is not a critical vertex: $R_1 - R_2 < C_{mrs}C_1$, then $MRS_{mrs} - RS_1 > 0$, which contradicts the assumption that C_{mrs} is the MRS center. □

Theorem 3 unveils the property that the circumscribed circle must pass through at least two vertices and the inscribed circle must be tangent to at least two edges of the polygon. See example 3.2.

4. COMPUTATION OF MRS CENTER

The medial axis MA(G) of a simple polygon G is composed of straight line segments and parabolic segments. A straight line segment is generated either by two boundary edges or by two vertices of the polygon. A parabolic segment is generated by a vertex and a boundary edge of the polygon.

If a medial axis segment spans through more than one farthest neighbor Voronoi polygon, it will be subdivided into subsegments denoted by MASS. Since each $MASS_i$ lies entirely in one farthest neighbor Voronoi polygon FNV(p_i), point p_i will be the farthest vertex of the $MASS_i$.

There is a radial separation (RS) value associated with each MASS. The value is determined by locating a specific point q in $MASS_i$ satisfying the following expression:

$$RS_{MASS_i} = \min_{q \in MASS_i} \left\{ d_2(q,p_i) - d_2(q,B_i) \right\}$$

where B_i is either the boundary edge or the vertex that generates the MASS. Since the locations of $MASS_i$, B_i, and p_i are three known factors, the location of point q can be computed analytically.

To locate the MRS center of a simple polygon G, the algorithm identifies the MASS with the smallest radial separation value:

1. Construct the medial axis MA(G) of the polygon.
2. Construct the farthest neighbor Voronoi diagram FNVOD(G) of the polygon.
3. Construct all the MASSs of the polygon by intersecting MA(G) with FNVOD(G).
4. Compute the RS value of each $MASS_i$. The point q of the MASS with the smallest RS value will be chosen as the MRS center of the polygon.

It should be noted that since an MRS center may not be unique, it is quite valid to have more than one MASS having the same smallest RS value. In this situation, any of these points can be chosen as the MRS center of the polygon.

Construction of the medial axis of a simple polygon with n vertices can be done in O(nlogn) time (see, e.g., [5,11]). The construction of the farthest neighbor Voronoi diagram of a set of n points can also be done within the same time bound (see [4,9]). The intersection of two sets of segments can be found in O(nlogn + k) time where n is the total number of segments and k is the number of intersections [1,7]. The computation of the RS value for each MASS can be done in O(1) time.

The algorithm can also be modified to compute the locus of the MRS center. For each MRS center found, the existence of its corresponding critical vertex p_* is checked. If one of the vertices on the circumscribed circle is the critical vertex of the MRS center, theorem 1 can be applied to determine the locus of the MRS center.

In the case of convex polygons, step 4 can be simplified by just computing the radial separation value of each intersection point and choose the one with the smallest radial separation value as the MRS center of the polygon.

We conclude the section with the following theorem.

THEOREM 4: The MRS center of a simple polygon G can be solved in O(nlogn +k) time, where n is the number of vertices of G and k the number of intersection of the medial axis and the farthest neighbor Voronoi diagram of G.

We note that the roundness problem studied by Ebara et al [3] is to compute the MRS center of a set of points. Although their algorithm has the same time complexity as ours, the two problems are considerably different. One is tempted to use the following approach to find the MRS center of a simple polygon. We first digitize the polygon boundary into sufficiently many points and then apply the algorithm presented in [3]. It turns out that the solution obtained by digitized points is not the true MRS center, as proved below. If the polygon of Example 3.1 is digitized into points (Example 4.1), the MRS center computed by the algorithm [3] will be located outside the boundary of the polygon. Not only the solution is incorrect, the introduction of digitization process will also increase significantly the size of the point set compared to that of the polygon.

5. MINIMUM AREA DIFFERENCE CENTER

In this section we discuss the minimum area difference (MAD) center. We first show the relationship between the MAD center and the MRS center of a polygon.

Let C_1 be the MAD center of a polygon G. The MAD value of C_1 is proportional to $R_2^2 - R_1^2$, where R_2 and R_1 are the radii of circumscribed and inscribed circle of the polygon:

$$MAD = \Pi\,(R_2^2 - R_1^2) = \Pi\,(R_2 + R_1)(R_2 - R_1)$$

Let M = $R_2 - R_1$. The MAD expression can be rewritten as:

$$MAD = \Pi\,(2R_1 + M)M$$

Let C_2 be the MRS center of the polygon with r_1 and r_2 being the inscribed and circumscribed radii of the polygon. Let m = $r_2 - r_1$. Then the area difference value defined by C_2 is:

$$AD = \Pi\,(2r_1 + m)m$$

In general, we have the following conditions:

$$m \leq M \tag{1}$$

$$R_1 \leq r_1 \tag{2}$$

$$R_2 \leq r_2 \tag{3}$$

$$MAD \leq AD \quad or \quad (R_1 + M)M \leq (r_1 + m)m \tag{4}$$

The condition (1) must hold since if M is smaller than m, the MAD center results in a smaller radial separation value than the MRS center. The condition (2) should be valid. Otherwise if r_1 is smaller than R_1, and since m is smaller than M according to condition (1), the MRS center yields a smaller area difference value than the MAD center. The same argument can be applied to condition (3). The condition (4) follows the definition of MAD center. Example 5.1 illustrates a specific case satisfying all aforementioned conditions. In this example, the MAD locus consists of the segment C_1C_2 and the MRS center is the end point C_1.

If M is equal to m, the intersection of MRS centers and MAD centers is non-empty, i.e. there exists at least a common point satisfying both the criteria of the MRS and MAD centers. Example 5.2 is a good contrast of the previous example. In this case, the solution of MRS center is the whole segment C_1C_2, and the MAD center is only the end point C_2 of the segment.

If equality holds in either condition (2) or (3), the same relation also holds in all four conditions, i.e., the solutions of the MRS center and the MAD center are identical. Indeed if R_1 is equal to r_1, then m has to be equal to M. Otherwise, if m is less than M, the area difference value of MAD center is larger than the AD value of the MRS center. Hence M = m would in turn imply the equality condition in both (3) and (4). The same arguments apply if the equality holds in condition (3). The point C_1 in example 5.3 satisfies both criteria of the MRS and MAD centers. Thus, if m is strictly less than M, then $R_1 < r_1$, and $R_2 < r_2$. Any polygon satisfying this condition will have distinct MRS and MAD centers. In example 5.4, the MRS center is C_1, and the MAD center is C_2, and m < M can be verified analytically.

The location of a MAD center is closely related to the FNVOD, and the *internal boundary*[(3)] of a polygon. This characteristic is stated by the following theorem.

THEOREM 5: The MAD center of a simple polygon must reside on a farthest neighbor Voronoi edge inside the polygon or on an internal boundary edge of the polygon.

PROOF: Suppose there exists a MAD center C_1 located inside a farthest neighbor Voronoi polygon FNV(p_1), but neither on any farthest neighbor Voronoi edge nor any internal boundary edge of the polygon. The MAD value of C_1 can be written as: $MAD = \Pi(C_1p_1^2 - r^2)$ where r is the radius of the inscribed circle. Imagine traversing along segment C_1p_1 from C_1, one either encounters a farthest neighbor Voronoi edge or an internal boundary edge. Let the first point encountered be denoted by C_2. Shifting the center from C_1 to C_2 will reduce the circumscribed radius by C_1C_2 and the inscribed radius by at most C_1C_2 in the worst arrangement. This worst case will occur only if segment C_1p_1 is perpendicular to the internal boundary of the polygon. (Note, the inscribed radius may increase, but this would further reduce the AD value.) The area difference value yielded by C_2 is:

$$AD = \Pi\left[(C_1p_1 - C_1C_2)^2 - (r - C_1C_2)^2\right] = \Pi\left[C_1p_1^2 - r^2 - 2C_1p_1C_1C_2 + 2rC_1C_2\right]$$

Since $C_1p_1 > $ r, the area difference value yielded by C_2 is smaller than the MAD value of C_1. This contradicts the assumption made. Hence the MAD center has to be either on a farthest neighbor Voronoi edge or an internal boundary of the polygon. □

The finding of the MAD center relies on the information of farthest neighbor Voronoi diagram and skeleton region elements SREs of the polygon. In order to compute the area difference value of each farthest neighbor Voronoi edge, the algorithm first identifies subsegments of farthest neighbor Voronoi edge, denoted by FNVESS, which lie in each SRE. In addition to that, internal boundary edges must be identified to locate possible MAD candidates. Each internal boundary edge may be divided by the farthest neighbor Voronoi polygons into more than one subsegment denoted by IBESS.

1. Identify all internal boundary edges (IBE) of the polygon
2. Construct the skeleton region elements SRE(G) of the polygon.
3. Construct the farthest neighbor Voronoi diagram FNVOD(G) of the polygon.
4. Construct the FNVESSs, and IBESSs of the polygon by intersecting FNVOD(G) with SRE(G). Identify only those FNVESSs inside the polygon.

---

(3) The internal boundary of a polygon includes all boundary edges of the polygon not forming the convex hull of the polygon.

5. Compute the Area difference value of each FNVESS, and IBESS to determine the MAD center of the polygon.

The complexity of the algorithm is dominated by that of step 4, which is O(nlogn + k) where k is the number of intersection points of FNVOD and SRE computed in step 2 and 3. Unlike the MRS algorithm, this algorithm will simultaneously locate the MAD center and its locus inside the polygon.

6. CONCLUDING REMARKS

In this paper, we have shown that one can compute the MRS center of a simple polygon from the medial axis and the farthest neighbor Voronoi diagram of the polygon. The algorithm takes O(nlogn + k) time where k is the number of intersection points of these two diagrams.

An analogous approach is used to compute the MAD center of a simple polygon. The MAD center of a simple polygon is derivable from the skeleton region elements and the farthest neighbor Voronoi diagram of the polygon. The time complexity of the algorithm is O(nlogn + k) where k is the number of intersection points of the skeleton region elements and the farthest neighbor Voronoi diagram of the polygon.

We list below a list of open problems. First, what is the number of intersections of medial axis and farthest neighbor Voronoi diagram of a simple polygon? A trivial upper bound is $O(n^2)$. Second, to find the MRS center can one locate the MASS which contains the solution without enumerating all possible intersections? Similar statements can be made for the MAD center problem. We conjecture that for convex polygons the MRS (or MAD) center can be found in O(nlogn) time. Third, what is the lower bound of the MRS and MAD center problem? Finally, unlike the practicality of the MRS center, which is justified in the introduction section of the paper, the application aspect of the MAD center remains to be explored.

REFERENCES

[1] B. Chazelle and H. Edelsbrunner, "An Optimal Algorithm for Intersecting Line Segments in the Plane", Proc. of 29th Annual Symp. on Foundations of computer science, Oct. 1988, pp. 590-600

[2] M.E. Dyer,"Linear Time Algorithms for two- and three- Variable Linear Programs", SIAM Journal on Computing, Vol 13, Feb. 84, pp. 31-45.

[3] H. Ebara et al, "Roundness Algorithms Using the Voronoi Diagrams", manuscript, submitted for publication.

[4] D.T. Lee, "Farthest Neighbor Voronoi Diagrams and Applications", #80-11-FC-04.

[5] D.T. Lee, "Medial Axis Transformation of a Planar Shape", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 4, July 1982.

[6] Samuel J. Levy, "Applied Geometric Tolerancing", Tolerances of Form Position, Proposed ISO/TC10/SC% R1101.

[7] H.G. Mairson and J. Stolfi, "Reporting and Counting Intersections between Two Sets of Line Segments", NATO Conference on Theoretical Foundations of Computer Graphics and CAD, July 1987.

[8] N. Megiddo, "Linear Time Algorithms for Linear Programming in R^3 and related problems", SIAM Journal on Computing, Vol 12.4, Nov. 1983, pp. 759-776.

[9] M.I. Shamos and D. Hoey, "Closest-Point Problems", Proc. 16th IEEE Symp. Foundations of Comput. Sci. Oct. 1975, pp. 151-162.

[10] G.T. Touissant, "Computing Largest Empty Circles with Location Constraints", International Journal of Computer and Information Science 12:5(1983), pp. 347-358.

[11] C.K. Yap, "An O(nlogn) Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments", Technical Report, Courant Institute, NYU, Oct. 1984.

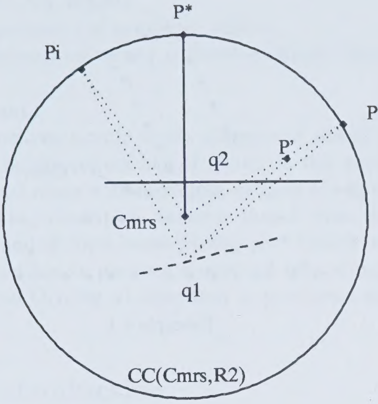

P*
Pi
P''
P'
q2
Cmrs
q1
CC(Cmrs,R2)

Figure 3.1

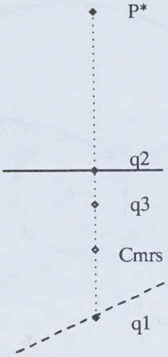

P*
q2
q3
Cmrs
q1

Figure 3.2

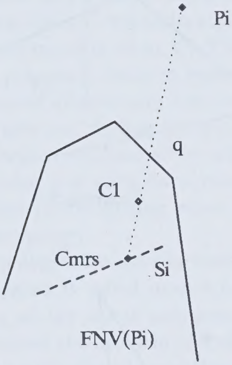
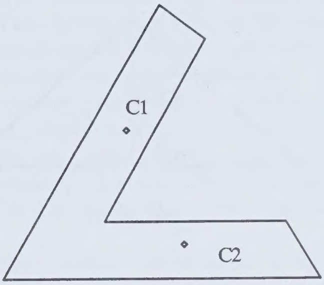
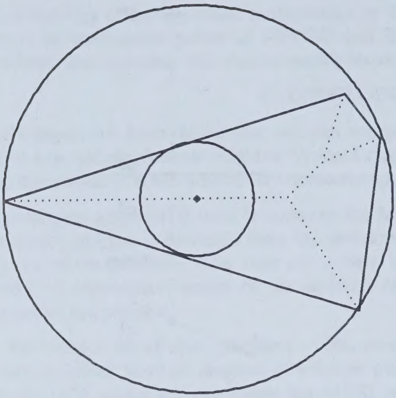

Pi
q
C1
Cmrs
Si
FNV(Pi)

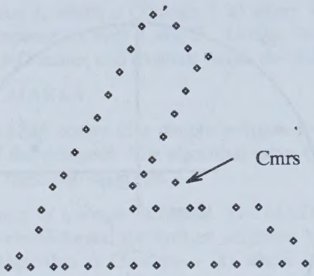
Figure 3.3



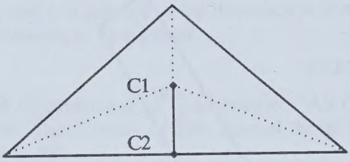
Example 3.1



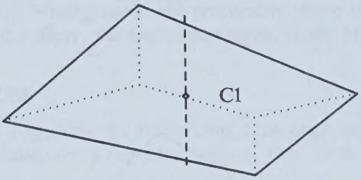
Example 3.2



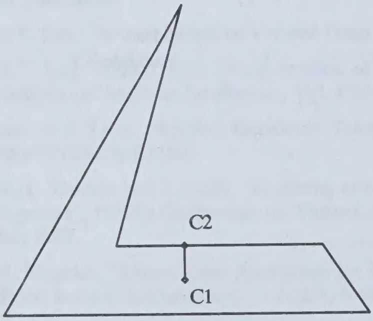
Example 4.1



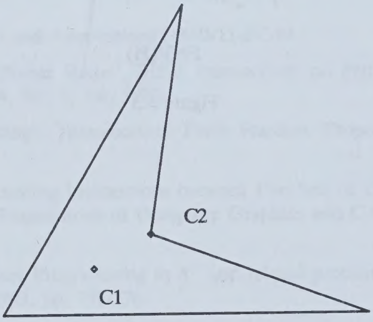
Example 5.1



Example 5.3



Example 5.2



Example 5.4

FINDING A MINIMAL CHAIN TO SEPARATE TWO POLYGONS

CAO AN WANG
Department of computer science
Memorial university of Newfoundland Canada A1C 5S7

Abstract

Given two simple n-side polygons P and Q in the plane, P is said to *contain* Q if no point of Q is on the convex hull of $P \cup Q$. In this paper, we propose efficient algorithms for finding a polygonal chain with minimum number of edges to separate P and Q (which is called *minimal separating chain*). In the case that P does not contain Q or in the case that P contains Q but P and Q form some *fixing port* (which will be defined later), we present an O(n *loglog* n) algorithm to produce a minimal separating chain. In the remaining cases of P and Q, we devise an O(n *log* n) algorithm to produce a minimal separating chain.

1 Introduction

Let C be a polygonal chain such that either C forms a polygon or the extreme segments of C are rays. Thus, C bi-partitions the plane. Let P and Q be two simple n-side polygons in the plane. Polygonal chain C is said to *separate* P and Q if the boundary of P lies on the opposite side of the boundary of Q along C. Therefore, if the boundaries of P and Q cross, then there cannot exist a polygonal chain to separate them, and we shall not consider this case.

The problem of determining a minimal separating chain for two polygons has many applications. Examples are: finding the minimum number of security persons to prevent prisoners escaping from one area (denoted by polygon P) into the other area (denoted by polygon Q). Finding a collision free path for a robot through a polygonal area such that the number of direction changes of the moving robot is minimal (which will reduce the computation load of an on-board computer).

When one polygon lies entirely inside the other polygon, a chain separating the boundaries of the two polygons is called *nested polygon*. If the number of edges of a nested polygon is minimal among all the nested polygons, it is called a *minimal nested polygon* . Aggarwal et al. [1] first investigated the problem of finding a minimal nested polygon for two convex polygons. They provided an O(n *log* n) algorithm for solving this problem. Suri and O'Rourke [9] derived an O(n^2) algorithm for solving the problem when the two polygons are simple.

In this paper, we shall find a minimal separating chain for two simple polygons in general position (refer Fig. 1.1 for four different positions of P and Q). We improve the previous results by deriving an O(n *loglog* n) algorithm and an O(n *log* n) algorithm to solve this problem. (Remark: Part of the result of this paper, the O(n *log* n) algorithm was presented in the conference paper [12].) Our O(n *loglog* n) algorithm is dominated by the time complexity of triangulating a simple polygon. If A triangulation of our desired simple polygon is given, our algorithm is worst-case optimal.

The main idea of our method can be described as follows. Given two polygons, the separating chains must be visible from some points of each polygon. The separating chains are thus determined by the 'facing parts' of the boundaries of the two polygons. (The *facing part* of the boundary of P with respect to Q is the shortest continuous piece of the boundary of P which contains all the vertices visible from some vertices of Q. The facing part of the boundary of Q with respect to P is defined similarly.) The facing parts determine a sequence of line segments, called a 'complete bridge' which connects the two facing parts. Our first observation is that the more segments of the complete bridge an edge of the separating chain crosses, the longer the edge is. Our second observation is if two polygons form some *fixing port* which anchors the separating chain, then a greedy method can be applied to minimalize the number of edges in the chain. Otherwise, a method of 'rotating calipers' is applied to produce a minimal separating chain after a preprocess of the two polygons.

The paper is organized as follows. Section 2 gives several definitions. Section 3 presents a method to find a complete bridge of two simple polygons, which will be used for finding a minimal separating chain. Section 4 proposes a method to find a minimal separating chain for two noncontaining simple polygons. Section 5 suggests a method to find a minimal nested polygon for two containing simple polygons with some fixing port. Section 6 sketches an algorithm to find the minimal nested polygon for two containing simple polygons with or without any fixing port. Section 7 concludes our works. For simplicity, we restrict ourselves only consider that the boundaries of two given polygons do not intersect.

2 Preliminaries

Definition: Let $\overline{xy}$ denote a line segment delimited by x and y. Let P denote a simple polygon, and U be its boundary. Line segment $\overline{xy}$ *does not cross* U if $\overline{xy}$-U does not lie on both sides of U. (Remark: Two line segments may intersect but do not cross.)

Definition: A point $x \epsilon P$ is *visible* from a point $y \epsilon Q$ if $\overline{xy}$ does not cross $U \cup V$, where P and Q are two simply polygons and U and V are their boundaries.

Definition: Let $CH(P \cup Q)$ be the convex hull of two simple polygons P and Q, P and Q are said to be *containing* if for all $x \epsilon P$, $x \not\epsilon CH(P \cup Q)$ or for all $y \epsilon Q$, $y \not\epsilon CH(P \cup Q)$. Otherwise, the two polygons are *noncontaining* . (Refer to case (1) and case (2) of Fig.1.1)

Definition: Given a sequence of line segments $(l_1,...,l_m)$, two line segments in the sequence are *adjacent* if they are next to each other in the sequence or they are l_1 and l_m.

Definition: Given two n-gons P and Q, a line segment $b = \overline{uv}$ for vertices $u \epsilon P$ and $v \epsilon Q$ is called a *bridge* if u and v are visible from each other. A sequence of line segments $B_r=(b_1,...,b_m)$ for $m \leq n$ is called a *sequence of bridges* , if every element of B_r is a bridge and if any two bridges of B_r do not cross. B_r is called a *complete bridge* of P and Q, if every line segment $\overline{xy}$ for points $x \epsilon U$ and $y \epsilon V$ intersects at least a bridge of B_r. (Thus, every two adjacent bridges of B_r, except b_1 and b_m of noncontaining case, share one of their endpoints.) (Refer to Fig.2.1 for a complete bridge.)

Definition: A polygonal chain C is called a *separating chain* of two polygons P and Q if P and Q lie on the opposite sides along C. Chain C is called a *minimal separating chain* if the number of edges in C is minimum among all possible separating chains of P and Q.

Thus, C is either a simple polygon or a simple chain with rays as its extremes. Moreover, C has the following properties. (1) none of the edges of C crosses the boundaries of P and Q, and (2) every edge of C intersects at least a bridge of B_r of P and Q and (3) each bridge of B_r intersects at most two edges of C .

3 Finding a Complete Bridge Between Two n-gons

In this section, we describe an O(n *loglog* n) procedure for finding a complete bridge. The complete bridge will be used for finding a minimal separating chain of the two n-gons.

Let $B_r=(b_1,...,b_m)$ be a complete bridge of two n-gons and let $C=(c_1,...,c_k)$ be a separating chain of the two n-gons. Each $c_i \epsilon C$ for $1 \leq i \leq k$ will intersect some bridge of B_r. Let the first and last bridges intersected by c_i be b_f and b_l respectively. Then, the first bridge b_f is *weakly visible* from the last bridge b_l [2] (that is, there exists at least a pair of points ($x \epsilon b_f$, $y \epsilon b_l$) such that $\overline{xy}$ does not cross the boundaries of the two n-gons). Using the weakly-visible property of the bridges of B_r, we can obtain a separating chain such that each edge of the chain will stretch as far as possible.

In the following, we shall describe how to construct a complete bridge of two n-gons P and Q. To do so, we first obtain the facing parts of the boundaries of P and Q. We then find a complete bridge from the facing parts. A complete bridge finally can be found by the following three-step process: (1) obtain the convex hull of P and Q. The convex hull will divide the plane into a closed part and an open part. The boundaries of P and Q and their convex hull will divide the closed part into several simple polygons, and/or a closed polygonal region. (2) identify one of these simple polygons (or the closed polygonal region), say R, such that the boundary of R contains the entire facing parts of the boundaries of P and Q. (3) obtain a complete bridge of P and Q from R.

Procedure CB(P,Q)

Step 1 Finding the convex hull of two n-gons.

Step 2 Locate the facing parts of the two n-gon.

The facing parts must lie inside the convex hull of P and Q due to their visibility property. In order to locate the starting segment for reporting the facing parts, we shall sequentially examine the edges of the convex hull. There are two cases, (a) and (b), as well as their subcases to consider.

(a) All the vertices of the convex hull belong to Q (for the case that all the vertices of the convex hull belong to P, the following analysis is applicable). Then, P is totally surrounded by Q. We shall further identify two subcases by testing if any vertex of P lies inside Q.

(i) If there exist such a vertex, then P must completely lie inside Q. (Fig.1.1 (a)(i))

(ii) Otherwise, P is totally surrounded by Q but P lies outside Q. (Fig.1.1 (a)(ii))

(b) There exist two convex hull edges, each of which spans a vertex in P and a vertex in Q. (It is easy to verify that there cannot exist only one such convex hull edge.) We shall also identify two subcases.

(i) If the two convex hull edges do not share any common vertex, then the facing parts must be contained in a simple polygon, where the simple polygon is bounded by the two convex hull edges and two facing parts of the boundaries of P and Q. (Fig.1.1 (b)(i))

(ii) Otherwise, the facing parts must be contained by a closed polygonal region (that is, a polygon with some of its vertices shared by four edges). (Fig.1.1 (b)(ii))

Obviously, a scan on the convex hull for identifying the two cases as well as their subcases takes at most $O(n)$ time. Thus, the facing parts can be found in $O(n)$ time.

Step 3 Finding a complete bridge from R.

To do so, we triangulate R using the method proposed by Tarjan and Van Wyk [10]; sequentially examine the triangles along the boundary of R and record all those edges, each of which spans a vertex previously belonging to P and a vertex belonging to Q. Hence, these recorded edges are bridges by definition. By rearranging the bridges according to their endpoints, we can always obtain a complete bridge, say B_r, in linear-time.

The above construction can be done in at most $O(n \textit{ loglog } n)$ time since triangulating R takes $O(n \textit{ loglog } n)$ time by Tarjan and Van Wyk's algorithm and since a scan on the edges of these triangles to form B_r takes $O(n)$ time.

We shall conclude the results of this section in the following two lemmas.

Lemma 3.1: Given two n-gons, a simple polygon or a closed polygonal region which contains the whole facing parts of the boundaries of the two n-gons can be constructed in $O(n)$ time.

Lemma 3.2: Given two simple n-gons, a complete bridge can be constructed in $O(n \textit{ loglog } n)$ time.

Now, the remaining problem is how to find a minimal separating chain of the two n-gons when a complete bridge B_r of the two n-gons is available. We shall first consider the noncontaining case and then consider the containing case in the following two sections.

4 Finding a Minimal Separating Chain for Two Noncontaining n-gons

The main idea of our method for constructing a minimal separating chain of two n-gons P and Q is based on the following two observations.

(1) Let R be the polygon containing a complete bridge B_r of P and Q. Let C be a minimal separating chain and let c_i be an edge of C. Let the first bridge and the last bridge crossed by c_i be b_f and b_l respectively. Then, b_l is weakly visible from b_f, and b_{l+1} is not visible from b_f. That is, c_i stretches to its maximal on R.

(2) The first bridge b_1 and the last bridge b_k of B_r must be crossed by a different edge of C respectively unless C consists of exactly one line segment.

Thus, a greedy algorithm can be designed to find a minimal separating chain C. The algorithm starts at b_1 to find the farthest bridge on B_r weakly visible from b_1, say b_l. Construct c_1 and find the farthest bridge on B_r weakly visible from c_1. Construct c_2 and find the farthest bridge weakly visible from c_2. Repeat the above process until b_k is reached. It can be shown that the resulting chain C is minimal.

The efficiency of the algorithm is relied on how fast to find the farthest bridge weakly visible from a given line segment (either b_1 or $c_i \epsilon C$). For convenience, we call the given line segment, *base segment*.

The base segment b and its farthest weakly visible bridge b_l delimited two pieces of the boundary of R. Let $U_{f,l}$ and $V_{f,l}$ denote the two pieces, where f is the index of the nearest bridge not crossed by b.

Definition: Let $U_{f,g}=(u_f,...,u_g)$ for $g \epsilon \{f,...,l+1\}$ be a piece of boundary U. The *internal convex hull* of $U_{f,g}$, denoted by $CH(U_{f,g})$, is a polygonal chain which shares the maximum number of vertices with $(u_i,...,u_g)$ and does not cross $(u_i,...,u_g)$, and each of whose vertex has a reflex angle in R.

Determining whether bridge b_g for $g \epsilon \{f+1,...,l+1\}$ is weakly visible from a base segment b can be done by testing the crossings between $CH(U_{f,g})$ and $CH(V_{f,g})$ and by testing if one of the two convex hulls lies totally inside the other convex hull. (For simplicity, we use testing the crossing between $CH(U_{f,g})$ and $CH(V_{f,g})$ to represent the above two testings.)

Note that the convex hull of $U_{f,l}$ is the convex hull of U' and U'' as shown in Fig. 4.1. (Similarly, $V_{f,l}$ may contain two pieces of the boundary V' and V'') It is not difficult to see that the vertices of $CH(U_{f,g})$ belonging to U' must also belong to $CH(U')$ and the vertices of $CH(U_{f,g})$ belonging to U'' must also belong to $CH(U'')$. Therefore, we shall first construct $CH(U')$ and $CH(V')$, then we construct $CH(U_{f,g})$ and $CH(V_{f,g})$ and test their crossing for each $g \ \epsilon \ \{f+1,..., l+1\}$.

We shall describe the construction of a minimal separating chain of two n-gons in the following three subsections. (4.1) construct $CH(U')$. (4.2) find the farthest bridge weakly visible from a base segment with respect to a given B_r. (4.3) present an algorithm for constructing a minimal separating chain of two n-gons.

4.1 Finding the convex hull of U'.

The following observation allows us to design a fast procedure to find the convex hull of U''. Suppose T is the triangulation of R we have obtained. Let c_i be a separating edge which we regard as a base segment. Let b_{f-1} be the last bridge crossed by c_i, and let u_r, u_s and u_t be the endpoint of b_{f-1}, the endpoint of c_i and the endpoint of b_f on U, accordingly. Then, $U' = (u_s, u_{s+1}, ..., u_t)$. Clearly, the vertices of $CH(U')$ must belong to those triangle edges of T which cross c_i and one of whose endpoints belongs to U'. To see this, note that each vertex of $CH(U')$ must have a reflex angle interior R. This implies there must exist some edges of T connect this vertex to some vertex not on U'. Note also that $\overline{u_r u_t}$ must belong to

T. Therefore, none of those edges of T, one of whose vertices belongs to U', can cross $\overline{u_r u_t}$. Then, each of these convex hull vertices must connect some vertex on chain $(u_r, ..., u_s)$ by an edge of T. Thus, these edges must cross c_i.

We call an angle $u_i u_j u_k$ for $1 \leq i < j < k \leq l$, an *open angle* of u_j if no vertex of chain $(u_r, ..., u_s)$ lies inside the area of the open angle. The *maximum* open angle of u_j is the largest one among all the open angles of u_j.

Procedure CH(U')

1 Sequentially examine the vertices of $U'=(u_s,...,u_t)$ and pick the vertices whose edges cross c_i. Let $(u_1, ..., u_l)$ denote the sequence of these vertices.

2 Sequentially examine the vertices of $(u_1, ..., u_l)$. Delete all those vertices that the maximum open angle of such a vertex is less than 90^o.

Lemma 4.1: Procedure **CH(U')** takes $O(n)$ time to produce the internal convex hull of U', where n is the number of vertices of U'.

Proof: Omitted.

4.2 Finding the farthest bridge weakly visible from a base segment.

Let b be the given base segment, b_f be the first bridge visible from b. (We assume if $b_f=b_1$, then $b=b_1$.) Suppose that we know the farthest bridge weakly visible from b is b_l for $l > f$. Then, b_{l+1} is not visible from b. Let $U_{f,g}$ and $V_{f,g}$ be the two pieces of the boundary of R delimited by the endpoints of b and b_g for $f \leq g \leq l+1$. Let '$\sqcap$' represent 'cross'. Then, $CH(U_{f,g}) \sqcap CH(V_{f,g}) = \emptyset$ if $g < l+1$ and $CH(U_{f,g+1}) \sqcap CH(V_{f,g+1}) \neq \emptyset$ if $g = l+1$.

The above observation indicates that we can determine the farthest bridge weakly visible from b by constructing $CH(U_{f,g})$ and $CH(V_{f,g})$ and testing the crossing of them for each g from $g=f$ to $g=l+1$. The efficiency of our method depends on (1) how fast can a pair of convex hulls be updated when a new bridge is taken into account and (2) how fast can a crossing of them be detected.

(1) When a new bridge is taken into account, only one of its vertices is added to the corresponding old convex hull. The other old convex hull remains no change. In order to construct a new convex hull from the old one, the procedure backtracks the vertices of the old convex hull until it reaches a vertex whose internal angle becomes reflex. Clearly, the new convex hull can be found in time proportional to the number of the vertices being backtracked.

(2) Now, we consider how fast can we test a crossing of $CH(U_{f,g})$ and $CH(V_{f,g})$ when we know $CH(U_{f,g-1})$ and $CH(V_{f,g-1})$ do not intersect. We shall show that the test can be done in amortized constant time. This claim depends on the following Lemma.

We first give a definition and describe the notations used in the Lemma.

Definition: Let $B_r = \{b_1,...b_s,...,b_m\}$ be a complete bridge of R of two n-gons, let c_i and c_j be two edges of a separating chain of the two n-gons. b_s and b_t be the last bridge crossed by c_i and c_j respectively. c_i is said to be *ahead* of c_j if (1) $s > t$ or (2) $t = s$ and for some $x \epsilon c_{j'}$ and $y \epsilon b_{s+1}$ $\overline{xy}$ intersects c_i and for all $z \epsilon c_{i'}$ and $y \epsilon b_{s+1}$ $\overline{zy}$ does not intersects c_j, where $c_{i'}$ and $c_{j'}$ are portion of c_i and c_j on the interval delimited by b_s and b_{s+1}.

Let $CH(U_{f,g})$ and $CH(V_{f,g})$ be the two convex hulls of $U_{f,g}$ and $V_{f,g}$ respectively. Let $CH(U_{f,g}) \sqcap CH(V_{f,g}) = \emptyset$ and let (x,y) be the closest pair of points between the two convex hulls. Thus, x and y divide the two convex hulls into two pairs of sub-convex hulls, $(CH^1(U_{f,g})$, $CH^1(V_{f,g}))$ and $(CH^2(U_{f,g})$, $CH^2(V_{f,g}))$, where the second pair lies ahead of the first pair in the given complete bridge. If the line extending $\overline{xy}$ intersects $CH(V_{f,g+1})$ say at z, then x and z will divide the new pair of convex hulls $CH(U_{f,g+1})$ and $CH(V_{f,g+1})$ into $(CH^1(U_{f,g+1})$, $CH^1(V_{f,g+1}))$ and $(CH^2(U_{f,g+1})$, $CH^2(V_{f,g+1}))$. (Refer to Fig. 4.2)

Lemma 4.2: (1) If $(CH(U_{f,g+1}) \sqcap CH(V_{f,g+1})) \neq \emptyset$, then $(CH^2(U_{f,g+1}) \sqcap CH^2(V_{f,g+1})) \neq \emptyset$, and (2) if $(CH(U_{f,g+1}) \sqcap CH(V_{f,g+1})) = \emptyset$, then the two closest points between $CH(U_{f,g+1})$ and $CH(V_{f,g+1})$ must lie on $CH^2(U_{f,g+1})$ and $CH^2(V_{f,g+1})$ respectively.

Proof: Omitted.

Part (1) of Lemma 4.2 implies that we can always detect the intersection of $CH(U_{f,g+1})$ and $CH(V_{f,g+1})$ by checking the intersection of $CH^2(U_{f,g+1})$ and $CH^2(V_{f,g+1})$. Part (2) of Lemma 4.2 implies that we can sequentially scan $CH^2(U_{f,g+1})$ and $CH^2(V_{f,g+1})$ starting at point y and at z respectively. We shall either detect an intersection of $CH(U_{f,g+1})$ and $CH(V_{f,g+1})$ (by part (1) of the lemma), or we shall find the closest pair of points between $CH(U_{f,g+1})$ and $CH(V_{f,g+1})$ (by part (2) of the lemma). Thus, testing of a crossing for all the pairs of the convex hulls (for $g = f$ to $l + 1$) takes linear-time.

The following procedure finds the farthest bridge visible from a base line segment and finds the separating edge c_{i+1}, where c_{i+1} is the ahead one of the two supporting line segments of the two last noncrossing convex hulls, and initially it is empty.

Procedure FaViBr (b, c_{i+1}, b_l).

```
1 $CH(U_{f,f}) \leftarrow$ CH($U'$); $CH(V_{f,f}) \leftarrow$ CH($V'$); $b_g \leftarrow b_{f+1}$;
2 While ($CH(U_{f,g}) \sqcap CH(V_{f,g}) = \emptyset$ and $b_{g+1} \neq \emptyset$) $g \leftarrow g+1$;
3 Find the ahead supporting line segment, $c_{i+1}$ and find the ahead piece $c'_{i+1}$;
    $b \leftarrow c'_{i+1}$; $l \leftarrow g$;
```

Lemma 4.3: Given a complete bridge of two n-gons and a base segment, the farthest bridge visible from the base segment can be found in linear-time by Procedure **FaViBr** , and the corresponding separating edge, in sub-linear time.

Proof: Omitted.

4.3 Constructing a minimal separating chain of P and Q.

We shall present an algorithm to construct a minimal separating chain of the two n-gons. In the algorithm, R is the polygon containing the facing parts of two noncontaining polygons P and Q, B_r is a complete bridge, C is the minimal separating chain of P and Q

Algorithm 1

```
Input : $P$ and $Q$
Output: $C$
Method:
1 Find $R$ from $P$ and $Q$;
2 Triangulate $R$ and form a complete bridge, $B_r = (b_1, ..., b_m)$ such that $b_1$ and $b_m$
    belong to the convex hull of $P$ and $Q$;
3 $b \leftarrow b_1$; $b_l \leftarrow b_2$; $i \leftarrow \emptyset$; $c_{i+1} \leftarrow \emptyset$;
    repeat
    FaViBr($b, c_{i+1}, b_l$);
    record $c_{i+1}$;
    $i \leftarrow i+1$;
    until $c_{i+1}$ crosses $b_m$
```

Theorem 1: Given two noncontaining n-gons, Algorithm 1 finds a minimal separating chain for them in O(n *loglog* n) time.

Proof: Omitted.

5 Finding The Minimal Nested polygon In Two Containing n-gons With Fixing Ports.

If P and Q are containing, then the area R, which contains a complete bridge of P and Q, is an annulus area. If we apply Algorithm 1 to P and Q, Algorithm 1 will start at some bridge which does not belong to $CH(P \cup Q)$. The first separating edge will extend to the both directions along the bridges. The output will be a simple polygon C. Moreover, the number of edges of C may not be minimal. This is because the first and the last edge of C may collapse into one edge after a shift of C on R. As matter of a fact, the number of edges in C is at most one more than that of a minimal nested polygon of the two n-gons. In other words, Algorithm 1 guarantees to produce a subminimal nested polygon of two containing n-gons.

Lemma 5.1: Let P contain Q and let $C=(c_1,...,c_k)$, produced by Algorithm 1, be a nested polygon of P and Q. Then, the number of edges of C is at most one more than the number of edges of a minimal nested polygon of the two n-gons.

Proof: Omitted.

Now, the question is whether or not we can find a minimal nested polygon of the two n-gons from a subminimal nested polygon C efficiently.

Definition: Let P contain Q. Let R denote the area containing a complete bridge of P and Q. A line segment is called a *spanning segment* if it lies in R and stretches to its maximal. A *fixing-port* of P and Q is (a) a sequence of three vertices ($v \epsilon Q$, $u \epsilon P$, $v' \epsilon Q$) which lie on the interior of a spanning segment in this order. (An edge $c \epsilon C$ containing the three vertices is called an *anchor*.) (b) a sequence of four vertices ($v \epsilon Q$, $u \epsilon P$, $u' \epsilon P$, $v' \epsilon Q$) such that $\overline{vv'}$ crosses the boundary of P and u and v lie on the interior of a spanning line segment and u' and v' lie on another spanning segment. (An edge $c \epsilon C$ containing one of (u, v) and (u', v') is called a *anchor*.) (Refer to Fig.5.1 and Fig. 5.2.)

It is easy to show that fixing ports of P and Q can be identified by examining the supporting vertices of C in linear-time. After identifying a fixing port (and corresponding anchor), we can apply Algorithm 1 again regarding the anchor as a base segment. It can be shown that the output of the algorithm is a minimal nested polygon.

Lemma 5.2: Let C be a subminimal k-gon of two containing n-gons produced by Algorithm 1. If the two polygon form fixing ports, then the minimalization of k-gon can be completed in $O(n \, loglog \, n)$ time.

Proof: Omitted.

6 Finding The Minimal Nested polygon In Two Containing n-gons Without Fixing Ports.

If the k-gon produced by Algorithm 1 contains no anchor, then we conclude that the k-gon is convex and its minimality can be achieved by a shift-test computation.

The idea is based on the 'rotating calipers' method. The efficience of the method is based on the following observation. When we shift the k-gon in the annulus area R, every instant of the shifting k-gon must be a convex polygon. Thus, the portion of the boundary of the shifting k-gon facing Q must be the convex hull of Q, and the portion of the boundary of the shifting k-gon facing P must be the maximum polygon which lies in P and which is visible from every point of Q (assume that only the boundary of P is regarded as an obstacle). We denote these two polygons by Q' and P'. Moreover, the vertices of the shifting k-gon must lie on P' and the edges of the shifting k-gon must be supported by Q'. Then, we can use a method similar to that presented in [1] to do the shift-test in $O(n \, log \, n)$ time. It can be shown that P' and

Q' can be constructed in O(n *log* n) time. We omitted the detail in this version and show the following lemma.

Lemma 6.1: Given two containing n-gons, if the k-gon produced by Algorithm 1 contains no anchor, a minimal nested m-gon can be found in O(n *log* n) time.

We summary the whole idea for finding a minimal nested polygon of two containing n-gons in the following algorithm.

Algorithm 2

```
Input:  $P$ and $Q$
Output:  $C$
Method:
1 Find a subminimal nested polygon $C'$ by Algorithm 1.
2 If $C'$ is obvious minimal, then $C'=C$ and Stop.
3 Sequentially scan on $C'$ to find anchors.
      If anchors occur Then
        find a minimal nested polygon $C$ by the method
        associated with Lemma 5.2
      Else
        find a minimal nested polygon $C$ by the method
        associated with Lemma 6.1
```

Theorem 2: Given two nonintersecting and containing n-gons, Algorithm 2 produces a minimal nested polygon in O(n *log* n) or O(n *loglog* n) time.

7 Concluding Remarks

We have presented an O(n *loglog* n) algorithm to find the minimal separating chain for two noncontaining n-gons. In the case that two containing polygons form some fixing port, our algorithm takes O(n *loglog* n) time to produce a minimal nested polygon. In the case that two containing polygons do not form any fixing port, our algorithm takes O(n *log* n) time to produce a minimal nested polygon.

The optimality of the first two algorithms depends on the discover of a linear-time algorithm for triangulating a simple polygon. To prove the last algorithm is optimal or show a improved time bound remains for further work.

REFERENCES

1. Aggarwal A., Booth H., O'Rourke J., Suri S., and Yap C., (1985), "Finding Minimal Convex Nested Polygons", *Proc. of The First ACM Symposium On Computational Geometry* , pp 296-304.

2. Avis D. and Toussaint G, (1981), "An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge", *IEEE Trans. Comput.* , Vol. c-30, pp 910-914.

3. Bhattacharya B. and EL Gindy H.,(1984) "A New Linear Convex Hull Algorithm for Simple Polygon", *IEEE Inform. Theory* , Vol. c-29, pp 571-573.

4. Edelsbrunner H., (1985),"On Computing The Extreme Distance Between Two Convex Polygons",*J. Algorithm* , Vol. 6, pp.213-224.

5. El Gindy H. and Avis D., (1981), "A Linear Algorithm for Computing the Visibility Polygon From a Point", *J. Algorithm* , Vol.2, pp.186-197.

6. Fournier A. and Montuno D. (1984)," "Triangulating Simple Polygons and Equivalent problems", *ACM Trans. on Graphics* , Vol. 3, pp. 153-174.

7. Hoffmann K., Mehlhorn K, Rosenstiehl, and Tarjan R., (1985), "Sorting Jordan sequences in Linear Time", *Proc. of The First ACM Symposium On Computational Geometry* , pp 196-203.

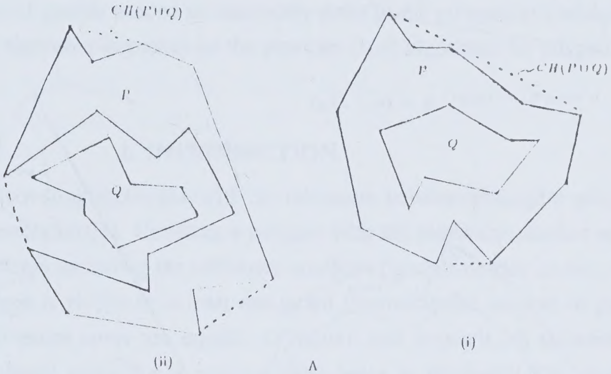
8. Lee D., Preparata F., (1984), "Computational Geometry - A Survey", *IEEE Trans. On Computers* , Vol. c-33, pp. 1072-1101.

9. Suri S. and O'Rourke J., (1985), "Finding Minimal Nested Polygons", *Technical Report* , The Johns Hopkins Univ..

10. Tarjan R. and Van Wyk C., (1988), "An O(n *loglog* n) algorithm for triangulating a simple polygon", *SIMA Journal on Computing* , **17** , pp.143-178.

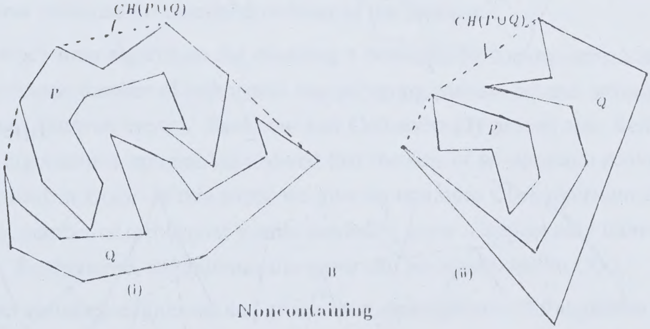
11. Toussaint G., (1982), "Complexity, Convexity, and Unimodality", *Proceedings of the Second World Conference on Mathematics* .

12. Wang C., Chan E., (1986), "Finding The Minimum Visible Vertex Distance Between Two Simple Polygons", *Proc. of The Second ACM Symposium On Computational Geometry* , pp 34-42.



A

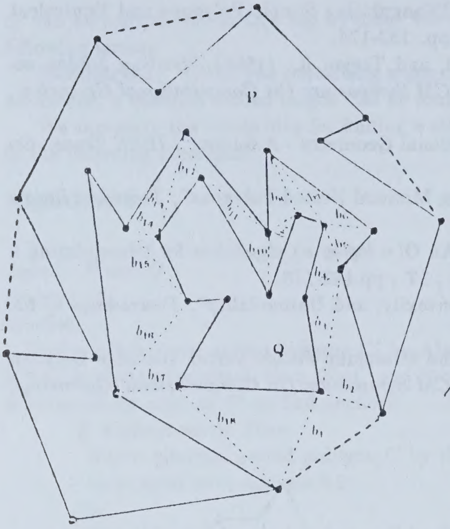
Containing



B

Noncontaining

Fig 1.1


P
Q
b_{11}
b_{12}
b_{13}
b_{14}
b_{15}
b_{16}
b_{17}
b_{18}
b_1
b_2
b_3
b_4
b_5
b_6

a complete bridge $B_r = (b_{11}, \ldots, b_{18})$

Fig 2.1

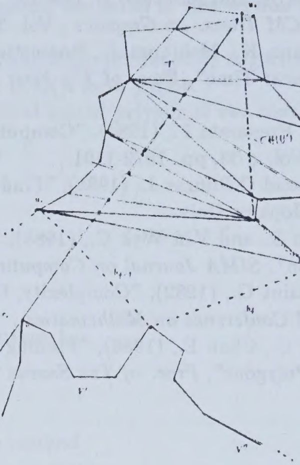

u
T
$CH(U')$
u
v''
b_i
v'
v''

Fig. 4.1

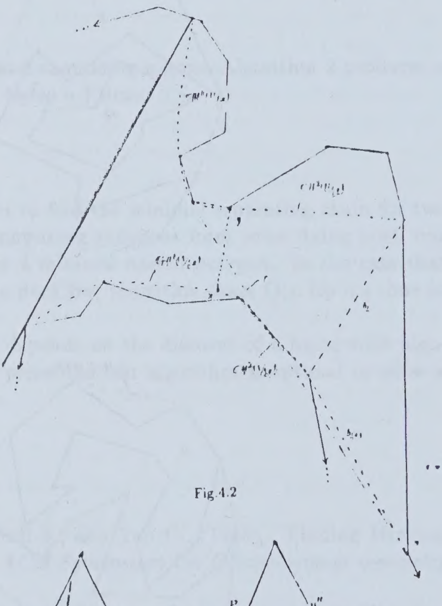
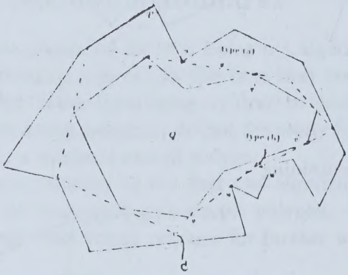
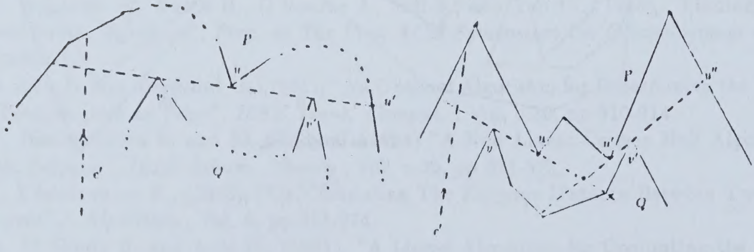

$CH^{(b)}(U_1)$
$CH^{(b)}(V_1)$
y
$CH^{(b)}(Q_1)$
$CH^{(b)}(V_2)$
b_1
b_{20}

Fig.4.2


P
type (a)
Q
type (b)
C

two types of fixing ports

Fig.5.1


P
u
u'
v'
Q
c'
P
u''
u
u'
v'
v
Q
c'

(a)

(b)

Fig.5.2

AN OPTIMUM ALGORITHM FOR COVERING A HORIZONTALLY CONVEX ORTHOGONAL POLYGON WITH ORTHOGONAL STAR-SHAPED POLYGONS

(Preliminary Version)

LAXMI GEWALI
Dept. of Computer Science and
Electrical Engineering
University of Nevada - Las Vegas
Las Vegas, NV 89154

SIMEON NTAFOS
Computer Science Program
The University of Texas at Dallas
Richardson, TX 75083-0688

Abstract: In this paper we present an optimum O(n) time algorithm for covering a horizontally convex orthogonal polygon with the minimum number of orthogonal star polygons. Finding minimum covers by star-shaped polygons is closely related to finding the minimum number of guards needed so that every point in the polygon is visible to at least one guard. Our algorithm improves on the previous $O(n^2)$ algorithm for this problem [3].

I. INTRODUCTION

The problem of covering a polygon with the minimum number of simpler polygons has attracted many researchers[5]. Covering a polygon with the minimum number of star polygons is closely related to finding the minimum number of guards needed so that every point inside the polygon is visible to at least one guard (the minimum number of guards and the size of a minimum cover are equal). O'Rourke and Supowit [6] showed that finding a minimum guard cover for a polygon with holes is NP-hard. The problem remains NP-hard even when the polygon contains no holes [1, 4]. In view of this, it is natural to seek efficient solutions for restricted versions of the problem.

Keil [3] gave $O(n^2)$ time algorithms for covering a horizontally convex orthogonal polygon with the minimum number of orthogonal star polygons (star cover) and orthogonally convex polygons (convex cover). Reckhow and Culberson [7] proved that Keil's $O(n^2)$ convex cover algorithm is optimal but showed that the size of an optimum convex cover can be determined in O(n). In this paper we give an optimum O(n) algorithm for finding the minimum number of orthogonal guards needed to cover a horizontally convex orthogonal polygon. Furthermore, an optimum star cover can be constructed in O(n).

In section II, we introduce definitions and give a short description of the algorithm in [3]. Keil's algorithm identifies pyramids corresponding to each horizontal line and uses the highest portions of the ceilings of these pyramids to determine the position of the guards. The algorithm takes O(n) to process each of O(n) pyramids, hence the O(n^2) complexity. In Section III, we show how to implement Keil's approach in O(n) by computing all the relevant information needed for the placement of the guards in a single sweep of the polygon and performing the actual placement in a second sweep. Unlike convex covers where two polygons in the cover can have an overlap of size O(n) (hence the (n^2) complexity [3,7]), the guard cover obtained by the algorithm is such that a star cover, of cardinality equal to the number of guards needed, can by computed in O(n) (the overlap between two star polygons in the cover is restricted to a rectangle).

II. KEIL'S ALGORITHM

Most of the terms defined in this section are similar to those used in [3]. An orthogonal polygon is called **horizontally convex** if its intersection with every horizontal line is either empty or a single line segment. An orthogonal polygon is called a **vertical histogram** if it has one horizontal edge, called the base edge, whose length is equal to the sum of the lengths of all the other horizontal edges. A **pyramid** is defined to be a horizontally convex vertical histogram. Unless otherwise stated, we use the term **polygon** to indicate a horizontally convex orthogonal polygon. A rectangle is called an **orthogonal rectangle** if its sides are parallel to the polygonal edges. Two points inside a polygon are said to be (orthogonally) **visible** if they can be placed inside an orthogonal rectangle contained in the polygon.

We are interested in designing an algorithm for stationing the minimum number of point guards inside a polygon so that each point in the polygon is visible to some guard. If we extend the edges of the polygon we get the **grid** induced by the polygon as shown in figure 1. The portion of a grid line lying inside the polygon is called a **grid segment**. The **level** of a horizontal grid segment is the number of horizontal grid segments lying on and below it. The grid segments partition the polygon into **rows** and **columns** of **rectangles**. Let D be any of the four compass directions, $\{N,S,W,E\}$, and let e be any edge of the polygon. If a vector normal to e and outward pointing from e is in the direction D then we say that e is a D_edge. The **ceiling** of a horizontal grid segment i is the set of portions of N_edges that are visible to i. The **top ceiling** of a horizontal grid segment i is the portion of the N_edge visible to i that has highest level. In figure 1, the dashed lines are the grid lines induced by the polygon. The shaded area is a pyramid. The thick segments form the ceiling of grid segment (a,b) and (c,d) is the top ceiling of grid segment (a,b).

Let $P(i)$ denote the region consisting of all rectangles in the first i rows of the polygon (from the bottom). Note that $P(i)$ is the region of the polygon below the $i{+}1^{th}$ horizontal grid segment. The level of the bottom most edge is 1. The algorithm performs a plane sweep from bottom to top and places guards one by one in such a way that when the sweep line is in the i^{th} horizontal grid segment, the minimum number of guards needed to cover $P(i)$ is determined. Let G_i denote the minimal guarding set for $P(i)$. Keil [3] computes G_i as follows.

Initially, G_0=*null*. At stage i, if G_{i-1} guards $P(i)$ then $G_i = G_{i-1}$. If not, $G_i = G_{i-1} \cup \{g\}$, where g is a new guard. The location of guard g is determined as follows: The i^{th} horizontal grid segment is called the **bottom boundary** of g. Let x' be the x_coordinate of any point belonging to the top ceiling of the i^{th} horizontal grid segment. Let j be the level of the highest horizontal grid segment such that when g is placed on j, at x-coordinate x', G_i guards $P(j-1)$. Such a horizontal grid segment is called the **top boundary** of g. Let y' be the y_coordinate of level j. Then, place g at the point (x', y'). In [3], it is shown that this strategy produces an optimum solution.

Keil implements the above strategy as follows. First the polygon is preprocessed to compute pyramids (and hence ceilings) corresponding to each horizontal grid segments. The pyramid of the i^{th} horizontal grid segment is determined by scanning the polygon above it. Computation of one pyramid takes $O(n)$ time [3]. Since there can be $O(n)$ candidates whose ceilings may have to be computed the total time complexity for computing all ceilings is $O(n^2)$ [3]. The algorithm then proceeds to perform a second sweep of the polygon and places guards by processing rows of rectangles one by one from bottom to top. The algorithm records whether or not 'top rectangles' (and thus all rectangles) in each column are guarded. Initially they are all unguarded. When the algorithm is processing row i it checks to see if G_{i-1} can see $P(i)$. Since G_{i-1} sees $P(i-1)$, G_{i-1} will also see $P(i-1)$ if top rectangles in every column passing through row i are visible (this is done by examining pyramid visible to the guard). If G_{i-1} guards $P(i)$ then G_i equals G_{i-1} and the next row is processed, otherwise a new guard, g, is placed and G_i is equal to $G_{i-1} \cup \{g\}$. The guard g is placed on a vertical grid line adjacent to a column passing through row i so that g achieves maximum ceiling. The guard g is placed on the highest grid line j such that G_i can guard $P(j-1)$. Once guard g is placed at level j the area visible to g above level j has to be marked. This is done by marking 'top rectangles' of columns visible from g in $O(n)$ time. This process is repeated at most $O(n)$ time. Thus the second sweep and hence the entire algorithm requires $O(n^2)$ time.

III. THE IMPROVED ALGORITHM

We use the same approach as in [3]. First, the polygon is preprocessed so that the top ceilings of all horizontal grid segments are found. Then, we determine the boundaries for each guard and place the guards. A horizontally convex orthogonal polygon's boundary consists of two chains, the **left chain** and the **right chain**, such that each chain is monotone with respect to the y_axis. Let h_i denote the level of the top ceiling of the i^{th} horizontal grid segment.

Lemma 1: Consider two horizontal grid segments at levels i and j having top ceilings at levels h_i and h_j respectively. If $j>i$ then $h_j \geq h_i$.

(Proofs are omitted in this extended abstract)

Let $elr_{i,j}$ denote the S_edge in the left chain lying between levels i and j and having its right end point rightmost among all such edges. Consider first elr_{i,h_i}, and let its level be $l1$. Next, consider elr_{l1,h_i} and let its level be $l2$. Continue like this until the edge immediately below level h_i in the left chain is reached. The set of horizontal edges, $LL_i=\{elr_{i,h_i}, elr_{l1,h_i}, etc.\}$, is called the set of **relevant edges in the left chain** between levels i and h_i. Similarly, we define $RR_i=\{erl_{i,h_i}, erl_{r1,h_i}, etc.\}$, to be the set of **relevant edges in the right chain** between levels i and h_i. In figure 2, we have that elr_{i,h_i} = (3,2), erl_{i,h_i} = (4,5). The thick edges in the figure are the relevant edges of the i^{th} grid segment.

Let xl_i, xr_i denote the x_coordinates of the left and right end points, respectively, of the top ceiling of the i^{th} horizontal grid segment. Then it is clear that:

$$\left.\begin{array}{l} xl_i=x\_coord(right\_end\_point(elr_{i,h_i})) \\ xr_i=x\_coord(left\_end\_point(erl_{i,h_i})) \end{array}\right\} \quad ...(1)$$

Let HL_i (resp. HR_i) denote the set of horizontal edges in the left chain (resp. right chain) that lie between i and h_i. The preprocessing proceeds by sweeping an infinite horizontal line from bottom to top to compute the top ceiling of each horizontal grid segment one by one. When the line sweeps from level i to h_i, the relevant edges between them are identified. The x-coordinates of the right end point (resp. left end point) of relevant edges in the left chain (resp. right chain) are maintained in linear list STL (resp. STR) during the sweep. STL,STR have two pointers, bottom and top, associated with each one. The top of each list works like a stack while the bottom works like a queue. Suppose that edges up to level i have been already processed. Assume that the $i+1^{th}$ edge belongs to the right chain. Then the x-coordinate of the left end point of the $i+1^{th}$ edge, x', is pushed into STR if x' is greater than the top of STR. Otherwise, STR is poped until either x' is greater than the top of SRT or STR is empty. Edges belonging to the left chain are similarly handled using STL. The edge e, currently swept, will contain the top ceiling of the i^{th} horizontal grid segment when either of the conditions "e belongs to HL_i and e overlaps in x-coordinate an edge in HR_i" or "e belongs to HR_i and e overlaps in x-coordinate an edge in HL_i" is satisfied. Two line segment $s1$ and $s2$ are said to **overlap in x-coordinate** if there are two points $p1$,$p2$ belonging to $s1$,$s2$, respectively, have the same x-coordinate. Note that the definition of "overlap in x-coordinate" is the same when one or both of $s1$,$s2$ is a point. Define "*condition* 1" and "*Condition* 2" as:

Condition1: $x\_coord(right\_end\_point(e))>Top(STL)$

Condition2: $x\_coord(left\_end\_point(e))<Top(STR)$

A formal algorithm for the sweep from i to h_i is given as Algorithm_1 in figure 3.

Lemma 2: When Algorithm_1 terminates, the edge e that is currently swept contains the top ceiling of the i^{th} horizontal grid segment.

Lemma 3: When Algorithm_1 terminates, STL (resp. STR) contains the x_coordinates of the right end points (resp. left end point) of all relevant edges in the left chain (resp. right chain) of the i^{th} horizontal grid segment.

The computation of the top ceiling of the i^{th} horizontal grid segment can be simply summarized as follows. Relevant edges of the i^{th} horizontal grid segment are first computed by sweeping the polygon from level i to h_i. The x-coordinate of the left end point (resp. right end point) of the top ceiling of the i^{th} horizontal grid segment is then given by the bottommost relevant edge in the left chain (resp. right chain). Our approach is to compute relevant edges of all horizontal grid segments and then compute top ceilings from them. Note that some of the relevant edges of the i^{th} horizontal grid segment are also the relevant edges of the $(i+1)^{th}$ horizontal grid segment and so on. Also, if any edge e is not a relevant edge for grid segment at level i then it is also not a relevant edge for any grid segment at level higher than i. We can therefore avoid repeated computation of common relevant edges by stacking them in STL and STR. This strategy makes it possible to compute all top ceilings in linear time.

Assume that the relevant edges between levels i and h_i are already in STL and STR. Without loss of generality assume that the $(i+1)^{th}$ horizontal edge belongs to the right chain. If the $(i+1)^{th}$ horizontal edge is also elr_{i,h_i} then the bottom pointer of STR is shifted up by one place so that the bottom most item in STR is now the x_coordinate of elr_{i+1,h_i}. Otherwise, elr_{i+1,h_i} is the same as elr_{i,h_i}. STL is updated in a similar fashion. Now, STL and STR contain the relevant edges between levels $i+1$ and h_i in the left and right chains, respectively. Next, the sweep line continues to sweep upward from h_i in exactly the same way as in Algorithm_1. This time, when Algorithm_1 terminates, the bottoms of STL and STR will have xl_{i+1} and xr_{i+1}, respectively, and the top ceiling of the $i+1^{th}$ grid segment has been found. This process is continued until the top ceilings of all horizontal grid segments are computed.

We can get the sorted list (according to y-coordinate) of horizontal edges of the polygon by simply merging the right and left chains in $O(n)$ time. The rest of the computation is charged to the horizontal edges encountered during the sweep. When the sweep line meets a horizontal edge, either a *Push* of one item or a *Pop* of several item is done. There can be at most n *Push* operations and, therefore, at most $O(n)$ *Pop* operations as well. Hence, the total charge to the horizontal edges is $O(n)$. We therefore have the following Lemma.

Lemma 4: The top ceilings of all horizontal grid segments induced in a horizontally convex orthogonal polygon can be computed in $O(n)$ time.

Next, we start a second sweep in which the boundaries of each guard are determined. The next two lemmas identify the bottom boundary and top boundary of each guard.

Lemma 5: Let $G_k = \{g_1, g_2, \cdots, g_r\}$ be the minimum guarding set for $P(k)$. Let l_1 (resp. l_2) be the level of the first N-edge above the k^{th} edge in the left chain (resp. right chain) having its right end point (resp. left end point) to the right (resp. left) of g_r. Let l_3 be the level of the first S_edge above the k^{th} edge. Then the level of the bottom boundary of g_{r+1} is the lowest level among l_1, l_2 and l_3.

Lemma 6: Let $G_k = \{g_1, g_2, \cdots, g_r\}$ be a minimum guarding set for $P(k)$. The top boundary of the next guard, g_{r+1}, is given by the first N_edge above the bottom boundary of g_{r+1} that is not completely visible to g_r.

Figure 4a illustrates the bottom boundary and top boundary of guards, where l_i and u_i represents, respectively, the bottom boundary and top boundary of the ith guard. Observe that l_3 is a S_edge and l_4 is a N_edge.

To find the boundaries for each guard, we sweep an infinite horizontal line from bottom to top. If the i^{th} and j^{th} horizontal edges are the bottom and top boundaries of g_r, let xl_i, xr_i be the x_coordinates of the left and right end points respectively of the top ceiling of the i^{th} grid segment. Then g_r is placed along j at x_coordinate xl_i (actually, any x_coordinate between xl_i and xr_i would do).

Assume that guards $g_1,...,g_r$ have been placed. As each guard is placed, Keil's algorithm [3] marks all edges above the guard that are visible to the guard (this requires O(n) time per guard). We process all the guards in O(n) by noting that it is enough to mark visible edges up to the top boundary of the next guard each time. This is so because, all edges that lie above the top boundary of g_{r+1} and are visible to g_r, are also visible to g_{r+1} and hence will be eventually marked. Instead of just marking the visible edges, we delete them during the sweep. The visibility of each edge from g_r is decided as follows. Any S_edge above g_r is not visible. During the sweep, the leftmost point in the right chain and the rightmost point in the left chain are updated. An N_edge, (a,b), is visible to g_r if the left most point in the right chain is not to the left of b and the rightmost point in the left chain is not to the right of a. During the sweep, the bottom and top boundaries of the next guard are determined using Lemmas 5 and 6, respectively. This information suffices to place the next guard and the process is continued until the whole polygon is swept. The formal algorithm is shown as Algorithm_2 in Figure 5. The algorithm performs O(1) operations per horizontal edge and we have:

Theorem 1: The minimum number of guards needed to see the whole of a horizontally convex orthogonal polygon can be determined in $O(n)$ time.

To find a minimum cover by star polygons in O(n) we have to construct star polygons with limited overlap so that the total size of the descriptions of all the polygons in the cover is O(n). We construct one star polygon for each guard in the optimum solution. Consider two guards g_i and g_{i+1} in the optimum solution. The star polygons for these two guards may have a rectangle R_{i+1} in common. This rectangle lies above l_{i+1} and is defined by two opposite corners a_{i+1}, b_{i+1}; one of the points is the inner vertex of the edge that formed l_{i+1} (e.g., point a_3 in figure 4b); the other point is the lower vertex of the last vertical edge in the edge sequence that is common to the boundary of the polygon and the pyramid associated with g_i (point b_3 in figure 4b). Then, the polygon associated with g_i includes the area between l_i and l_{i+1} plus the rectangles R_i, R_{i+1}, plus the portion of the pyramid associated with g_i that is bounded by l_{i+1} and R_{i+1}, minus the area bounded by l_i and R_i (this has been included in the previous star polygon). Note that one or both of the rectangles R_i, R_{i+1} may have zero area (e.g., R_2, R_4 in figure 4b). Since the overlap between star polygons has size O(1), a minimum star cover can be computed and described in O(n).

Acknowledgements

Simeon Ntafos is supported in part by a grant from Texas Instruments.
Laxmi Gewali is supported in part from U.S. Army Office under grant DAAL03-87-G-004.

References

1. Aggarwal, A., "The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects", Ph.D. Thesis, Department of Computer Science, John Hopkins University, Baltimore, Maryland, 1984.
2. Culberson, J. and K. Supowit, "Covering Polygons is Hard", 29th Annual Symposium on Foundations of Computer Science, Oct. 1988.
3. Keil, J.M., "Minimally Covering a Horizontally Convex Orthogonal Polygon", Proceeding of the Second Annual Symposium on Computational Geometry, June 1986, pp. 43-51.
4. Lee, D.T. and Lin, A., "Computational Complexity of Art Gallery Problems", IEEE Tranc. on Info. Theory, Vol. IT-29, No.2, March 1986, pp. 276-282.
5. O'Rourke, J., "Art Gallery Theorems and Algorithms", Oxford University Press, 1988.
6. O'Rourke, J. and K. Supowit, "Some NP-hard Polygon Decomposition Problems", IEEE Trans. on Info. Theory, vol. IT-29, No.2, March 1983, pp. 181-190.
7. Reckhow, R. A. and Joseph Culberson, "Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons", Proc.of the Third Annual Symposium on Computational Geometry, Canada, 1987, pp. 268-277.

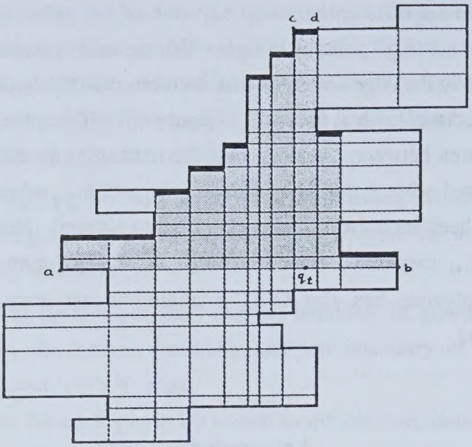


Figure 1

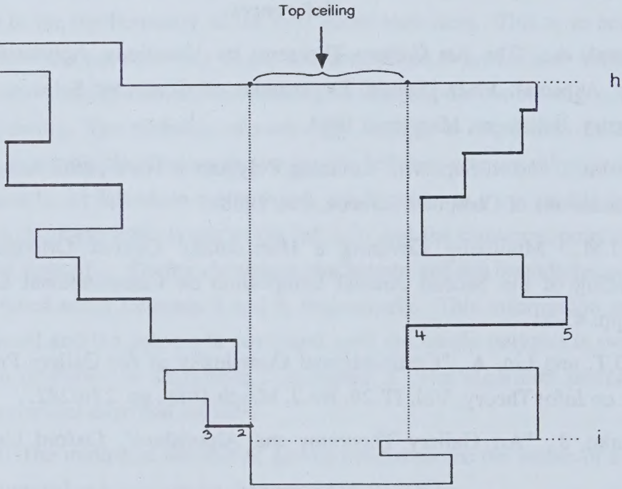

Top ceiling
h_i
4
5
3
2
i

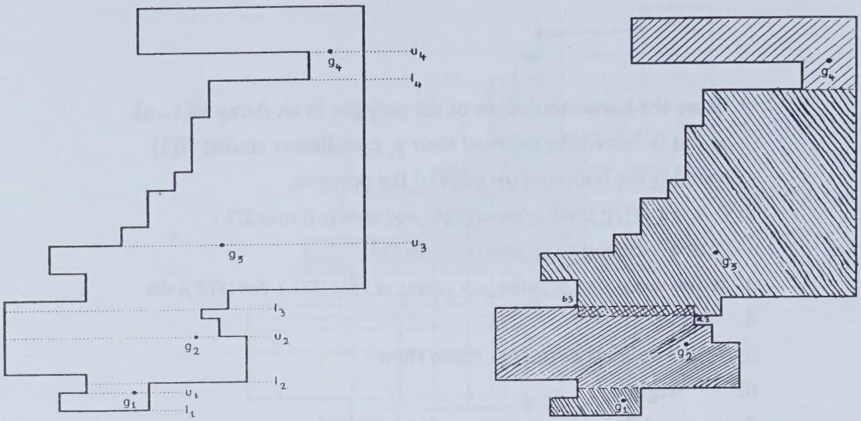
Figure 2

```
Store the horizontal edges of the polygon in an Array H[1...n],
  sorted in increasing order of their y_coordinates so that H[1]
  contains the bottom most edge of the polygon.
$j \leftarrow i$; $e \leftarrow H[j]$; Push $x\_coord(right\_end\_point(e))$ into $STL$;
  Push $x\_coord(left\_end\_point(e))$ into $STR$;
while $sweep\_line\_is\_below\_top\_ceiling\ (e, Top(STL), Top(STR))$ do
  begin
    if e belongs to the left chain then
      begin
        while $not\ (empty(STL)\ or\ Condition\,1)$ do
          $Pop\ STL$;
        $Push\ (x\_coord(right\_end\_point(e))\ into\ STL$;
      end
    else
      begin
        while $not\ (empty(STR)\ or\ Condition\,2)$ do
          $Pop\ (STR)$;
        $Push\ x\_coord(left\_end\_point(e))\ into\ STR$;
      end;
    $j \leftarrow j+1; e \leftarrow H[j]$;
  End;
$xl \leftarrow Get\_bottom\_item(STL); xr \leftarrow Get\_bottom\_item(STR)$;   $y \leftarrow y\_coord(e)$;
  $left\_point \leftarrow (xl, y)$;   $right\_point \leftarrow (xr, y)$;   $h_i \leftarrow (left\_point, right\_point)$;
```

The function $sweep\_line\_is\_below\_top\_ceiling$ is defined as:

```
$sweep\_line\_is\_below\_top\_ceiling(e, x1, x2)$;
  if (e ε left chain) and (x2 < x_coord(right_end_point(e)))
        or
     (e ε right chain) and (x1 > x_coord(left_end_point(e)) then
        return(true)
  else $return(false)$;
```

Figure 3: Algorithm_1.



(a) (b)

Figure 4

```
Let $LC$ and $RC$ be the list of horizontal edges in the left chain and right chain respectively sorted in the increasing order of their y_coordinates.
$i \leftarrow 1$ {bottom boundary}; $j \leftarrow$ level of the bottom most N_edge {top boundary}
Delete edges in $LC$ and $RC$ up to $j^{th}$ horizontal edge.
while there are edges in $LC$ or $RC$ do
  begin
    $t \leftarrow$ top ceiling of $i^{th}$ grid segment.
    $xl_i \leftarrow$ x_coordinate of the left end point of $t$.
    $xr_i \leftarrow$ x_coordinate of the right end point of $t$.
    $y \leftarrow$ y_coordinate of $j$.
    Place guard at point $(xl_i, y)$.
    Walk along $LC$ and $RC$ and delete edges up to the top boundary of the next guard. Record the right most point in the left chain and the left most point in the right chain during the walk.
    Set $i$ and $j$ the level of the bottom and top boundary, respectively, of the next guard by using Lemma 5 and 6 during the walk.
  end
```

Figure 5: Algorithm_2.

A LOOP-FREE SPANNING-TREE PROTOCOL IN DYNAMIC TOPOLOGY

CHUNHSIANG CHENG, and *SRIKANTA P. R. KUMAR*
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

INTRODUCTION

Spanning-tree protocols are important in network management, as they provide a minimum structure connecting all the nodes. It can be applied for broadcasting using the minimum number of messages, and loop-free routing. In this paper, we present a new resilient spanning-tree protocol for the communication networks represented by undirected graphs. Given an initial spanning tree, the protocol responds to arbitrary link/node failures by reconstructing a spanning tree when it is broken, and link/node recoveries by reconnecting the components. Assuming that a finite number of topological changes occur at arbitrary times during a period (and no more changes after this period), the protocol will obtain a spanning tree corresponding to the latest network within a finite time after the last topological change (with standard link layer assumptions, see [2,3]).

Given m link failures and k recoveries, the protocol uses O(kn+me) messages, each with O(logn) bits where n and e are the maximum number of nodes and links respectively throughout the computation. In contrast, if one applies any fixed topology spanning-tree protocol P on top of the global or local reset protocols (suggested by Segall [3], and Afek, Awerbuch, and Gafni [1]) to reconstruct the spanning-tree after every failure or recovery, the message complexity will be O((m+k)*(x)), with O(logn) bits/messages, where x is the message complexity of P, which appears to be O(e) as the best known message complexity.

The presented protocol applies the dynamic fragment extension idea by using the node identity as the fragment identity such that a fragment with bigger identity can only be invited by another fragment with smaller identity. This idea is derived from the minimum spanning tree protocol [2], which needs unbounded counter to generate fragment identities. The need of unbounded counter is eliminated in this context of spanning tree protocol. Also, the presented protocol is loop free in the sense that the subgraph represented by the union of the selected tree-links is without loops at any instant during the computation.

A SPANNING-TREE PROTOCOL

A fragment at an instant is a rooted spanning tree of a component of the network at that moment. For each tree link (u,v) form node u to node v, we call u the parent node of v, and v the child of u. Th root is the only node that has no parent. We assign for each fragment a fragment identity which is stored by each node to classify its adjacent links into two classes: the links connecting to nodes having the same fragment identity, called ingoing links; and the links connecting to the nodes having different fragment identity, called outgoing links. Fragment extension involves searching for an outgoing link to connect to another fragment. The subgraph resulted from connecting more than one fragment will broadcast a new fragment identity to all the nodes in it such that this subgraph becomes a new fragment and can recursively do the next fragment extension (see also [2,4]).

The spanning-tree protocol derives broadcast-with-echo phases on tree links to do the fragment extension. For a fragment with identity f, the root will first determine if there is any link connecting outwardly to any other fragment with smaller fragment identity. If there is a such outgoing link (u,v) such that node u is located inside the fragment f, and v in another fragment f' and f'<f, then node u will become the root and connect to f'. At the time that node v accept the connection, fragment f will become part of f'. We describe the distributed implementation of this idea as follows.

For each fragment, the computation starts from the root of this fragment by sending a FIND message to each of its children through its adjacent tree links, and changes its state to be *find*. Any node receiving FIND message from its parent will respond as the root by sending each child a FIND message, and changing its state to be *find*. The execution of a node in *find* state is to find a outgoing edge if there is any. This is accomplished by sending TEST message, argumented with its fragment identity, to its neighbors except its parent and children. Upon receiving a TEST<f> message from a neighbor, a node will return an ACCEPT message when its fragment identity is smaller than f, or REJECT message otherwise. A link of a node will be marked as outgoing edge when through which ACCEPT message is received. A node which has children and is in *find* state will send the FIND_ACK to its parent only when it has received the FIND_ACK from all its children. Each FIND_ACK is also argumented. When a node u finds at least one outgoing edge, the FIND_ACK message is argumented as FIND_ACK<u>; or, it is argumented as FIND_ACK<x> when any FIND_ACK<x> is received from a child. Otherwise, FIND_ACK is sent as FIND_ACK<null>.

When the root receives all the FIND_ACK from all its children, it will send CHANGE_ROOT to a neighbor returning a FIND_ACK<u> where u is a node name. This CHANGE_ROOT<u> message will be transmitted to node u through the tree links and reverse the parent-child relation of the links in the path from the original root to node u. This is accomplished by letting each node x, upon receiving CHANGE_ROOT<u> from parent--say, y-- where u is not x, mark y as a child, send CHANGE_ROOT<u> to the child--say, z--, and mark node z as the parent . If a node u receives CHANGE_ROOT<u>, then it will send CONNECT<f> to a neighbor v such that link (u,v) is marked as an outgoing edge where f is the fragment identity stored at node u. Upon receiving this CONNECT<f> message, node v will check again if its fragment--say f'-- is smaller than f. If f'<f, then node v will send MERGE<f'> to node u and mark u as a child. Else node v returns NOCONNECT to node u. Receiving MERGE<f'>, node u will first broadcast to the nodes originally in fragment f the new fragment identity f'. This can be done by broadcast-with-echo phase. After this reidentification task is done, node u will do whatever should be done as a child of node v. Receiving NOCONNECT, node u will use a broadcast-with-echo phase to reidentify fragment f with the new fragment identity u and start over again from finding an outgoing edge.

When finding the outgoing edge, if the root receives FIND_ACK<null> from all children, a SLEEP message is sent to return all nodes to *normal* state. Upon receiving a SLEEP message from the parent, a node simply goes into *normal* state.

When the failure on any link (u,v) is detected by node u, node u responds as receiving a NOCONNECT message at node u if node v is the parent of node u. When the recovery on any link (u,v) is detected by node u, if node u is in *normal* state, it will send a TEST<f> on to link (u,v)where f is the fragment identity stored at node u. If the returned message is ACCEPT, then node u will send WAKEUP message through parent toward the root to start over from finding an outgoing edge. The protocol is a modification of that algorithm in detail in [2].

Reference

[1] Y. Afek, B. Awerbuch, and E.M. Gafni, "Applying Static Network Protocols to Dynamic Networks,"*Proc. 28th IEEE Symp. on Found. of Comp. Sci.*. pp. 358-370, Oct 1987.

[2] C. Cheng, I.A. Cimet, and S.P.R. Kumar, "A Protocol to Maintain Spanning Tree in a Dynamic Topology," *ACM SIGCOMM Symp. Commun. Arch. and Protocols*, pp. 330-338, Stanford, CA, Aug. 1988.

[3] A. Segall, "Distributed Network Protocols," *IEEE Trans. Inf. Theory IT-29*, pp. 23-35, Jan. 1983.

[4] R.G. Gallager, P.A. Humblet, and P.M. Spira, "A distributed Algorithm for MST", *ACM Trans. Prog. Lang. Syst.*, 5: 66-67, 1983.

A FAST TOPOLOGY MAINTENANCE ALGORITHM FOR HIGH-BANDWIDTH NETWORKS

HOSAME ABU-AMARA
Texas A & M University
Department of Electrical Engineering
College Station, Texas 77843

ABSTRACT

A *topology change* in a network of host computers is a link or a host crashing, or a link or a host resuming operations. *Topology maintenance* is the problem of preserving a correct view of the network at each host when the network changes its topology. We present the fastest time-driven algorithm that we know of for topology maintenance in high-bandwidth networks. Our algorithm uses only 4 time units for each broadcast by each host. The best previous algorithm required $O(\log n)$ time units for each broadcast by each host, where n is the number of hosts in the network. One advantage of our algorithm is that we do not require a host to broadcast its messages to all its neighbors in the same step.

1. INTRODUCTION

A *topology change* in a network of host computers is a link or a host crashing, or a link or a host coming up. *Topology maintenance* is the problem of preserving a correct view of the network at each host when the network changes its topology. A correct view of the network is important for route computation, congestion control, and fault-tolerance.

In traditional networks, processing capacity dominates transmission capacity. Several topology maintenance algorithms are known for traditional networks. In such algorithms, the hosts repeatedly broadcast their topology information to other hosts. The algorithms can be divided into two classes: event-driven and time-driven. In an *event*-driven algorithm, a host broadcasts its topology information *only when* the host detects a topology change. The algorithm of Tajibnapis [5] is an example of event-driven topology maintenance algorithms. Although Tajibnapis does not present a running time for his algorithm, he states that the number of messages his algorithm uses is exponential in the number of hosts in the network and is linear in the number of topology changes. In a *time*-driven topology maintenance algorithm, the hosts periodically broadcast their topology information regardless of whether any topology change actually occurs. The ARPANET algorithm of McQuillan et al [4] is an example of time-driven algorithms. Let e be the number of links in the network, and n be the number of hosts. Each broadcast by each host in the ARPANET algorithm requires $O(e)$ messages and runs in $O(n)$ time units.

Only recently have algorithms for high-bandwidth networks been proposed. In such networks, transmission capacity dominates processing capacity [1, 3, 6]. Using a novel model for such networks, Cidon et al [2] propose a time-driven topology maintenance algorithm that requires $O(n)$ messages and $O(\log n)$ time units for each broadcast by each host.

In our paper, we improve on Cidon et al's algorithm by presenting a time-driven topology maintenance algorithm that requires $O(n)$ messages but only 4 time units for each broadcast by each host. Another advantage of our algorithm is that although Cidon et al allow hosts to broadcast their messages to all their neighbors in the same step, we do not use this broadcast mechanism in our algorithm.

2. MODEL

Our algorithm is designed for high-bandwidth networks. In these networks, transmission and switching capacity dominate processing capacity. Thus we follow the novel model proposed by Cidon et al [2]. The communication network is represented by a graph, where each node in the graph represents a host, and each link represents a bidirectional communication channel. In what follows, we will not distinguish between nodes and hosts, nor between links and channels. Let n be the number of nodes in the graph. Each node consists of a *switching subsystem* and a *node control unit.* The switching subsystem of a node v receives each message M, sends a copy of M to v 's node control unit, and, depending on the header of M, either discards M or forwards M to another node. The node control unit of v extracts the useful information from the copy of M, does some internal computations, and sends a message to v 's switching subsystem to be forwarded to another node.

Following Cidon et al's model [2], we assume that there are no link delays because transmission capacity dominates processing capacity. As in Cidon et al, however, we restrict the number of link traversals to be at most n for each message. When we compute the running time of the algorithm, we suppose that generating a message requires at most one time unit. (See Cidon et al's paper for a justification of these assumptions.)

Each link and node is either *active* or *inactive* at various times. A link is *active* at time t if the link does not lose any message that is on the link at time t. If a link is not active, then it is *inactive.* An inactive node is a node all of whose links are inactive. We will follow the standard assumption that a node adjacent to an inactive link knows that the link is inactive [2, 4, 5].

A *topology change* is a node or a link changing its active/inactive state. In our model, each node maintains a graph of the node's connected component. We will present a time-driven topology maintenance algorithm such that if all topological changes cease at some time, then every active node will eventually have the correct graph of the node's connected component.

Before we present our algorithm, we need the following definitions.

3. DEFINITIONS

Let T be a tree of n nodes. Then $level(v,T)$ of every node v in T is recursively defined as follows: If v is the root of T, then $level(v,T) = 0$; if v is not the root of T, then $level(v,T) = 1 + level(parent\ of\ v, T)$. The *depth* of a tree is the biggest level among all the nodes in the tree. Let u and v be two nodes in T. A *common ancestor* of u and v is a node w such that u and v are descendent from w in T. The *closest common ancestor* of u and v is the node $cca(u,v)$ that has the biggest level among all the common ancestors of u and v. When we say "visit the nodes $(x_1, x_2, \ldots, x_y)$ in T ", we mean: Follow the shortest path in T from x_1 to $cca(x_1, x_2)$; then follow the shortest path in T from $cca(x_1, x_2)$ to x_2; then follow the shortest path in T from x_2 to $cca(x_2, x_3)$; ...; then follow the shortest path in T from $cca(x_{(y-1)}, x_y)$ to x_y.

3.1 *Incremental traversal of T*

In each level of T, order the nodes from left to right. Let $u_{(1,i)}$ denote the leftmost node in level i in T, and $u_{(k,i)}$ denote the k-th leftmost node in level i. Thus $u_{(1,0)}$ is the root of T. Let $N(i)$ be the number of nodes in level i, and T be of depth D. Then, we define the *incremental traversal* of T as follows: For $i = 1, 2, \ldots, D$, follow the shortest

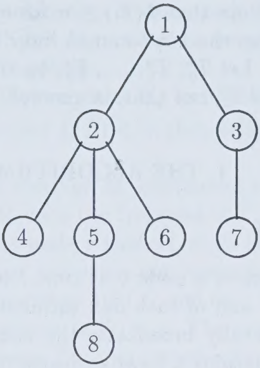


Figure 1: The Tree T_E

path from $u_{(1,0)}$ to $u_{(1,i)}$; visit ($u_{(1,i)}, u_{(2,i)}, \ldots, u_{(N(i),i)}$); and follow the shortest path from $u_{(N(i),i)}$ to $u_{(1,0)}$.

Let *counter*(T) be a counter that counts the number of link traversals when an incremental traversal is done on T. It is easy to show that if T is of depth D, then *counter*$(T) \leq 2\sum_{r=1}^{D}\sum_{i=1}^{r} N(i)$. Consider, for example, the tree T_E in Figure 1. The tree is traversed in an incremental fashion by visiting the nodes in the following order: 1, 2, 1, 3, 1, 2, 4, 2, 5, 2, 6, 2, 1, 3, 7, 3, 1, 2, 5, 8, 5, 2, 1.Thus, *counter*(T_E) is 22 links when the traversal of T_E ends.

3.2 *Tree decomposition*

The *decomposition* of T is the finite sequence *DECOMPOSE(T)* of subtrees T_0, T_1, ..., T_θ, defined recursively as follows:

(1) Suppose that r_0 is the root of T. Then T_0 is the subtree rooted at r_0 obtained by traversing T in an incremental fashion until *counter(T)* = *n*. (The value of n is fixed throughout the recursion. If r_0 does not have enough descendants for *counter(T)* to be n, then T_0 is the subtree rooted at r_0 that contains r_0 and all the descendants of r_0.) Define the *length* $L(T_0)$ of T_0 to be the value of *counter(T)* when T_0 was created.

(2) Let r_1, r_2, ..., r_l, be a sequence of nodes in T_0 ; a node r is in the sequence if, and only if, at least one child of r is not in T_0. The node r_i precedes r_j in the node sequence if $level(r_i, T) > level(r_j, T)$, or if $level(r_i, T) = level(r_j, T)$ but r_i is to the left of r_j. For each r_i in the node sequence, let $\tilde{T}_i$ be the subtree of T rooted at r_i that contains: r_i , the children of r_i that are not in T_0, and all the descendents of these children. Recursivley decompose the subtrees $\tilde{T}_1$, $\tilde{T}_2$, ..., and $\tilde{T}_l$.

(3) The decomposition of T is the sequence *DECOMPOSE(T)* = T_0, *DECOMPOSE(* $\tilde{T}_1$ *)*, ..., *DECOMPOSE(* $\tilde{T}_l$ *)*.

For example, the decomposition of T_E is the sequence of subtrees (1,2,3,4,5), (5,8), (2,6), (3,7) ; the lengths are 8, 2, 2, and 2 respectively. We call each T_i, $0 \leq i \leq \theta$, a *decomposition subtree* of T. Note that $L(T_i) \leq n$ for every $0 \leq i \leq \theta$, and that no two distinct decomposition subtrees share a common link. If $L(T_i) = n$, then we also call T_i a *full* decomposition subtree. Let $T_0^\star$, $T_1^\star$, ..., $T_\beta^\star$, be the sequence of *full* decomposition subtrees of T. Note that $T_0^\star = T_0$, but that, in general, $T_i^\star$ is not necessarily the same as T_i.

4. THE ALGORITHM

4.1 *Overview of the algorithm*

We define the *local topology* of a node v at time t to be the active/inactive state, at time t, of each neighbor of v and of each link incident to v. As in ARPANET [4] and PARIS [2], each node periodically broadcasts the node's current local topology to all other nodes. Every node maintains a local sequence number that the node increments every period. Let z be a node. To keep the time required to broadcast z 's local topology a constant, we use the following three techniques:

(1) Each broadcast by z uses a shortest path tree rooted at z.

(2) Let T be a shortest path tree rooted at z, and let M be a message sent by z to traverse T. In the algorithm of Cidon et al [2], M can not visit a node v in T if M already visited one of v 's children. Cidon et al call this property *one-way traversal*, and they show that $\Omega(\log n)$ time units are required to broadcast z 's local topology in one-way broadcast algorithms. We do not require tree traversals to be one-way in our algorithm.

(3) Once a node generates a message, we maximize the number of nodes the message visits before the message is discarded.

Unlike Cidon et al, we do not allow a node to broadcast messages to all the node's neighbors in the same step. Note that traversing the shortest path trees in a depth first fashion may not work since the traversing message may be lost on encountering the first inactive link or node.

4.2 *Details*

Each node z periodically executes the following:

Step 1: The node z constructs a shortest path tree T for the network by using z 's current idea of the network topology. The tree T is rooted at z. Let T_0, T_1, ..., T_θ, be the decomposition of T. Node z sends a message M_z, containing T and z 's local topology, to traverse in an incremental fashion the hosts and the channels that correspond to T_0. (M_z also contains the current sequence number to distinguish M_z from the messages sent by z in previous periods.) If M_z attempts to visit an inactive host or cross an inactive link, then M_z is discarded. When M_z ends the traversal of T_0, M_z is discarded.

Recall that $u_{(j,i)}$ denotes the j-th leftmost node in level i in T. The switching subsystem of every node $u_{(j,i)}$ that M_z visits forwards a copy of M_z to $u_{(j,i)}$'s control unit. When the control unit of $u_{(j,i)}$ receives M_z for the first time, $u_{(j,i)}$ does the following. Recall that M_z contains T. The node $u_{(j,i)}$ computes the integer $p = j + \sum_{k=0}^{i-1} N(k)$, where $N(k)$ is the number of nodes in level k in T. Also, $u_{(j,i)}$ may execute Steps 2 and 3(b):

Step 2: If the number of full decomposition subtrees of T is less than p, then $u_{(j,i)}$ does not execute Step 2. Otherwise, $u_{(j,i)}$ spawns a message M_u that contains a copy of M_z. The message M_u follows the shortest path in T from $u_{(j,i)}$ to the root of $T_p^\star$, and M_u traverses $T_p^\star$ in an incremental fashion. When M_u ends the traversal of $T_p^\star$, the message

M_u is discarded. The switching subsystem of every node v that M_u visits forwards a copy of M_z to v 's control unit. Since M_u contains a copy of M_z, every node that receives M_u will know z 's local topology.

Suppose that $\hat{v}$ is a node in $T_p^\star$ that receives M_u for the first time. If $\hat{v}$ is the root of a decomposition subtree $\hat{T}$ of T, where $L(\hat{T}) < n$, then $\hat{v}$ executes Step 3(a). Otherwise, $\hat{v}$ does nothing.

Step 3(a): The node $\hat{v}$ spawns a message $\hat{M}$, containing a copy of M_z, to traverse $\hat{T}$ in an incremental fashion. When $\hat{M}$ ends the traversal of $\hat{T}$, the message $\hat{M}$ is discarded. The switching subsystem of every node v that $\hat{M}$ visits forwards a copy of M_z to v 's control unit. The message $\hat{M}$ does not cause any messages to be spawned.

Step 3(b): The node $u_{(j,i)}$ executes Step 3(b) if, and only if, $u_{(j,i)}$ is the root of a decomposition subtree $T\prime$ of T, where $L(T\prime) < n$. The node $u_{(j,i)}$ spawns a message $M\prime$, containing a copy of M_z, to traverse $T\prime$ in an incremental fashion. When $M\prime$ ends the traversal of $T\prime$, the message $M\prime$ is discarded. The switching subsystem of every node v that $M\prime$ visits forwards a copy of M_z to v 's control unit. The message $M\prime$ does not cause any messages to be spawned.

5. PROOF OF CORRECTNESS

We follow the definition of correctness proposed by Tajibnapis [5] and Cidon et al [2] for topology maintenance algorithms. Thus, suppose that all topological changes cease at some time t_f. We will show that, eventually, every node in the network will know the correct topology of the node's connected component.

According to our algorithm, all nodes periodically broadcast their local topology information. Let z be a node that wishes to broadcast z 's local topology at some time $t_z > t_f$, and let T be the shortest path tree that z constructs at time t_z.

Lemma 1: Suppose that $\tilde{T}$ is a subtree of T with $L(\tilde{T}) = n$. Then $\tilde{T}$ contains at least $\sqrt{n} - 1$ bidirectional links.

Proof: Recall that, if $\tilde{T}$ is of depth D, then $L(\tilde{T}) \leq 2\sum_{r=1}^{D}\sum_{i=1}^{r}\tilde{N}(i)$, where $\tilde{N}(i)$ is the number of nodes u in $\tilde{T}$ with $level(u, \tilde{T}) = i$. Thus,

$$n = L(\tilde{T}) \leq 2\sum_{r=1}^{D}\sum_{i=1}^{r}\tilde{N}(i)$$

Note that

$$2\sum_{r=1}^{D}\sum_{i=1}^{r}\tilde{N}(i) \leq \sum_{i=1}^{D}\tilde{N}(i)\left[\sum_{j=1}^{D}\tilde{N}(j) + 1\right]$$

and that $\sum_{i=1}^{D}\tilde{N}(i)$ is the number of links in $\tilde{T}$. Let $\sum_{i=1}^{D}\tilde{N}(i) = N(\tilde{T})$. Hence

$$\begin{aligned} n &\leq N(\tilde{T})\left[N(\tilde{T}) + 1\right] \\ N(\tilde{T}) &\geq \frac{-1 + \sqrt{1 + 4n}}{2} > -1 + \sqrt{n} \end{aligned}$$

Thus $\tilde{T}$ contains at least $\sqrt{n} - 1$ links.□

The following lemma shows that a message M that traverses a decomposition subtree visits a node in level j for the first time only if M did not visit any node in level $j + 1$.

Lemma 2: Let T_p be a decomposition subtree for T, and let M be a message spawned to traverse T_p in an incremental fashion. Suppose that M visits some node v_p in T_p for the first time at some time t_v. Let $NODES(M, t_v)$ denote the set of all nodes that M visited before time t_v. Then $level(u, T) \leq level(v_p, T)$ for every $u \in NODES(M, t_v)$.
Proof: The message M was originally spawned by either the root r_p of T_p or by a node w_p outside T_p.

If M was spawned by r_p, then the lemma is true by the definition of incremental traversal.

If M was spawned by w_p, then, by our algorithm, w_p is in T_0. Thus, $level(w_p, T) \leq level(s, T)$ for every node s in T_p. Recall that $cca(w_p, r_p)$ is the closest common ancestor of w_p and r_p. By the algorithm, M travels the path P_1 from w_p to $cca(w_p, r_p)$, and then travels the path P_2 from $cca(w_p, r_p)$ to r_p. Again, note that $level(u, T)$ of every node u on $P_1 \cup P_2$ is at most $level(s, T)$, for every node s in T_p. In particular, $level(u, T)$ of every node u on $P_1 \cup P_2$ is at most $level(v_p, T)$. When M traverses T_p in an incremental fashion, the lemma holds true by the description of incremental traversal.□

Lemma 3: Suppose that T has a level i such that the following two properties hold for every node v with $level(v, T) \leq i$: (1) the host corresponding to v is active after t_f, and (2) the channel corresponding to the link between v and the parent of v in T is active after t_f. Then, at some time after t_z, every node v with $level(v, T) \leq i$ will know the local topology of z.
Proof: Let v be in the decomposition subtree $T(v)$. If $T(v) = T_0$, then the message M_z that z spawns to traverse T_0 in Step 1 of the algorithm will visit v *before* M_z visits any node in level $i + 1$, by Lemma 2. Hence Lemma 3 will be true.

If $T(v) \neq T_0$, then either $L(T(v)) = n$ or $L(T(v)) < n$. If $L(T(v)) = n$, then Lemma 1 proves that every decomposition subtree $\tilde{T}$ with $L(\tilde{T}) = n$ contains at least $\sqrt{n} - 1$ links. Hence T_0 contains at least $\sqrt{n} - 1$ links. Thus there are at least $\sqrt{n}$ nodes that execute Step 2 of the algorithm. Since there are at most $n - 1$ links in T, and since no two distinct full decomposition subtrees share a common link, there are at most $\sqrt{n}$ full decomposition subtrees. Thus, there are enough messages spawned in Step 2 to traverse each full decomposition subtree. In particular, there is a message spawned to traverse $T(v)$. By Lemma 2, Lemma 3 will be true.

Finally, if $L(T(v)) < n$, then the root of $T(v)$ is in a full decomposition subtree. Thus the root of $T(v)$ spawns a message to traverse $T(v)$ in Step 3 of the algorithm. By Lemma 2, Lemma 3 will be true.□

Theorem 1: Eventually, every node in the network will know the correct topology of its connected component.
Proof: Recall that all topological changes cease at some time t_f. Let G_f be the network graph after time t_f. We define $d(u, v)$ to be the number of links in the shortest path in G_f between nodes u and v. Let $l(w, v)$ denote a link in G_f that connects a node w to a node v. For all nodes u, v, and w in G_f, let $G_f(u, i)$ denote the following subgraph of G_f: node v is in $G_f(u, i)$ if, and only if, $d(u, v) \leq i$; the link $l(w, v)$ is in $G_f(u, i)$ if, and only if, w and v are in $G_f(u, i)$ and $d(u, w) + d(u, v) \leq 2i - 1$. We prove Theorem 1 by showing that for each node u and each $i \geq 1$ there exists a time after which u knows the correct topology of $G_f(u, i)$. The proof proceeds by induction on i.
Base: $i = 1$. In our model, each node u always knows the the local topology of u. Hence, u always knows the correct topology of $G_f(u, 1)$.

Inductive Step: Suppose that for each u there exists a time after which u knows the correct topology of $G_f(u,i)$. Fix a particular node u_0 in G_f. We will show that there exists a time after which u_0 knows the correct topology of $G_f(u_0, i+1)$.

Suppose that u_1 is a node such that $d(u_0, u_1) = i$. By the inductive hypothesis, there exists a time $t(u_1, i) \geq t_f$ after which u_1 knows the correct topology of $G_f(u_1, i)$. According to the algorithm, u_1 periodically broadcasts $G_f(u_1, 1)$. Consider what happens when u_1 broadcasts after $t(u_1, i)$. Since u_1 knows the correct topology of $G_f(u_1, i)$, the broadcast tree T that u_1 constructs has the property that, for every node v with $level(v,T) \leq i$, v is active and the link in T between v and the parent of v is active. Since $d(u_0, u_1) = i$, the node u_0 will correctly know the local topology of u_1 by Lemma 3. Thus u_0 will know the correct topology of $G_f(u_0, i+1)$.□

Let d be the network diameter. The proof of Theorem 1 shows that $O(d)$ broadcasts per node may be needed before all nodes know the correct topology of the network. The topology algorithm of Cidon et al [2] also requires $O(d)$ broadcasts per node.

6. TIME ANALYSIS

Theorem 2: In each broadcast by each node, the algorithm uses at most 4 time units.

Proof: Recall that, in our model, there are no link delays, and that generating a message requires at most one time unit. Suppose that a node z wishes to broadcast z 's local topology by using a broadcast tree T. In Step 1 of the algorithm, M_z traverses at most n links. Hence Step 1 requires at most one time unit. In Step 2, the message M from each node $u_{(j,i)}$ has to travel to the root of the full decomposition subtree $T_p^\star$ before M can traverse $T_p^\star$, where $p = j + \sum_{k=0}^{i-1} N(k)$. Since there are at most $n-1$ links in the longest path in T, then M requires at most one time unit to travel to the root of $T_p^\star$, and M requires at most one time unit to traverse $T_p^\star$. Hence Step 2 requires at most 2 time units. Finally, each message in Step 3 traverses at most n links. Hence, Step 3 requires at most one time unit.□

ACKNOWLEDGEMENT

My thanks go to Inder Gopal for many helpful discussions.

REFERENCES

[1] I. Cidon and I. Gopal, "PARIS: An approach to private integrated networks," *Journal of Analog and Digital Cabled Systems*, to appear.

[2] I. Cidon, I. Gopal, and S. Kutten, "New models and algorithms for future networks," *manuscript*; also, a preliminary version in *Proc. 7th ACM Symp. Principles Distributed Comput.*, Toronto, Canada, Aug. 1988.

[3] J. Y. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE Journal on Selected Area in Commun.*, Vol. SAC-5, No. 8, pp. 1264-1273, Oct. 1987.

[4] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *IEEE Transactions on Communications*, Vol. COM-28, pp. 711-719, May 1980.

[5] W. D. Tajibnapis, "A correctness proof of a topology information maintenance protocol for a distributed computer network," *Communications of the ACM*, Vol. 20, No. 7, pp. 477-485, July 1977.

[6] J. S. Turner and L. F. Wyatt, "A packet network for integrated services," *Proc. Globecom'83, IEEE Global Telecomunications Conference*, pp. 45-50, Nov. 1983.

Data Redistribution Scheduling in Fully Connected Networks

Ravi Varadarajan, Pedro I. Rivera-Vega and **Shamkant B. Navathe**
Computer and Information Sciences Department
University of Florida, Gainesville, Florida

Abstract

We show that the problem of scheduling file redistribution in fully connected networks is NP-hard even with the restriction that no file uses more than two edges in its route. We provide an efficient algorithm for constructing a schedule from an optimal or approximate solution that only specifies the number of files that use each two-edge route. We also discuss the special case when there is only a single source or a single destination.

1. Introduction

In this paper we consider the problem of scheduling the redistribution of files in a computer network which is *fully connected*, that is, between any two nodes, there are two communication links, one in each direction. In this problem, the initial locations of files as well as their destinations are known beforehand. Every file takes constant time (without loss of generality assume unit time) to move along any link. Each link can be used to transfer only one file during any time unit and every node in the network can simultaneously send as well as receive as many files as the number of outgoing and incoming links. A file can be moved in a *store and forward* fashion along a route from its source to its destination. We impose no restriction on the storage capacity of the node. We also ignore the processing costs of the nodes in directing the transfer of files; this is a reasonable assumption since the complete schedule is determined beforehand and the nodes do negligible processing during the transfer. The objective is to find a schedule so as to minimize the completion time of the transfer referred to as the *makespan* of the schedule. The schedule includes for each file a route as well as the times for usage of the links.

The problem is of practical importance in distributed and parallel computer systems. For example, in a distributed database system, redistribution of data is periodically required in order to maintain a desired level of performance usually measured by average execution time of transactions. In this case the redistribution should be accomplished as soon as possible since the system cannot be fully operational while it is taking place. In the parallel processing context, this problem is related to the problem of routing packets among the processors of a network ([7]). The difference between Valiant's model and our problem is that in our model, the scheduling is determined prior to the actual transfer while scheduling is not considered separately in the other model. In this paper, we do not address the issue of how the schedule is efficiently transmitted to the different nodes of the network.

In a related problem, referred to as *File Transfer Problem* (FTP) ([2,3]), the objective is to find a schedule with minimum makespan for the transfer of files in a network but with the constraint that each file transfer requires communication modules at both ends; there is a limited number of such modules at each node and they are not dedicated to communication links. In almost all versions of the file transfer problem forwarding of files is not allowed. Our problem can be reduced to an FTP in which each communication link has two dedicated communication modules, a transmitter at one end and a receiver at the other; we also allow forwarding of files.

We are interested in constructing *simple* schedules so that each node needs to maintain a small amount of information to direct the transfer. Specifically we attempt to find optimal *two-edge* schedules in which no file needs to use more than *two* edges in its route from its source to its destination.

This paper can be summarized as follows. In section 2 we introduce the notations and prove that our problem is NP-hard. In section 3 we present a method to exactly solve the problem. In section 4 we study the particular case when there is only one source (destination) and one or more destinations (sources). We present the conclusions in section 5.

2. Problem Description and Complexity

File Redistribution Problem

A *redistribution model* M is a tuple (V, E, S, D, F) where

$G = (V, E)$ is a directed graph, where V is the set of vertices representing the nodes of the network and E the set of edges representing unidirectional communication links. In our problem, $E = \{(u,v) | u, v \in V \text{ and } u \neq v\}$.

F - set of *files* that need to be reallocated, $F = \{f_1, f_2, \ldots, f_m\}$.

S - source function, $S : F \longmapsto V$ where $S(f_i)$ indicates the node where file f_i is initially located.

D - destination function, $D : F \longmapsto V$ where $D(f_i)$ indicates the node to which f_i has to be transferred.

We say that a file f is of **type [i,j]** if $S(f) = i$ and $D(f) = j$. Any schedule is represented by a function $\mathcal{S} : E \times \mathcal{N} \rightarrow F \cup \{\epsilon\}$, where $\mathcal{N}$ is the set of natural numbers; it is defined as follows:

$$\mathcal{S}(e,t) = \begin{cases} f & \text{if file } f \text{ uses link } e \text{ at time } t \\ \epsilon & \text{if no file uses link } e \text{ at } t \end{cases}$$

For each file f, let $t_f(\mathcal{S})$ be the time taken by f to move from its source to its destination in the schedule $\mathcal{S}$; $t_f(\mathcal{S}) = \min\{t+1 | \mathcal{S}(e,t) = f, \text{ where } e \text{ is an incoming edge of } D(f)\}$. The makespan of the schedule $\mathcal{S}$, denoted by $T(\mathcal{S})$, is given by $\max_{f \in F} t_f(\mathcal{S})$. For a given redistribution model M, the **file redistribution problem** is one of determining a schedule with minimum makespan. We denote the optimum makespan for M by $T_{\text{opt}}(M)$.

In a *two-edge* schedule, each file uses at most two edges in its route from its source to its destination. We are interested in finding *two-edge schedules* with minimum makespan. We conjecture that there is always an optimal two-edge schedule for the file redistribution problem. We are currently attempting to prove or disprove this conjecture, but so far we could only corroborate it in many particular cases.

Problem Complexity

In this section we prove the intractability of the file distribution problem in fully connected networks by transforming the Satisfiability Problem (SAT) ([4]) to an instance of this problem. Our approach is to first construct a redistribution model which we call as M_1 for an instance of the SAT problem; in this model, the graph is not fully connected. We then transform M_1 to M_2 in which the graph is fully connected in such a way that both models have the same optimal makespan.

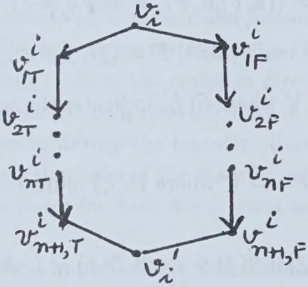
In a given instance of the SAT problem, let C_1,C_2,...,C_n be the clauses in the boolean expression Q, and let $\{u_1, u_2, \ldots, u_m\}$ be the set of variables used. In the corresponding redistribution model $M_1 = (V_1, E_1, S_1, D_1, F_1)$ the graph (V_1, E_1) is constructed as follows:

1. For each u_i there is a subgraph $G_i' = (V_i', E_i')$ where

   $$V_i' = \{v_i, v_i', v_{1T}^i, v_{1F}^i, v_{2T}^i, v_{2F}^i, \ldots, v_{nT}^i, v_{nF}^i, v_{n+1,T}^i, v_{n+1,F}^i\},$$

   $$E_i' = \{(v_i, v_{1T}^i), (v_i, v_{1F}^i)\} \bigcup \{(v_{jT}^i, v_{j+1,T}^i), (v_{jF}^i, v_{j+1,F}^i) | j = 1, 2, \ldots, n\} \bigcup \{(v_{n+1,T}^i, v_i'), (v_{n+1,F}^i, v_i')\}$$

   Note that the edges $(v_{jT}^i, v_{j+1,T}^i)$ and $(v_{jF}^i, v_{j+1,F}^i)$ are introduced for each clause C_j in the boolean expression. G_i' has the following form:



2. For each clause C_j introduce nodes w_j and w_j'. For every literal $u_i \in C_j$ ($\neg u_i \in C_j$) introduce vertices and edges so as to create a path from w_j to v_{jF}^i (from w_j to v_{jT}^i) of length i; also introduce vertices and edges so as to create a path from $v_{j+1,F}^j$ to w_j' (from $v_{j+1,T}^j$ to w_j') of length $n - i + 1$. Thus for each literal in C_j, we have created edge disjoint paths from w_j to w_j' of length $n + 2$.

The file set F_1 of M_1 is composed of two file sets defined as follows:

- $U = \{f_{u_i} | i = 1, 2, \ldots, m\}$; $S(f_{u_i}) = v_i$ and $D(f_{u_i}) = v_i'$ for $i = 1, 2, \ldots, m$.
- $C = \{f_{c_j} | j = 1, 2, \ldots, n\}$; $S(f_{c_j}) = w_j$ and $D(f_{c_j}) = w_j'$ for $j = 1, 2, \ldots, n$.

Note that for each of the $m+n$ files, the length of any path from its source to its destination is $n+2$. For $j = 1, 2, \ldots, n$, there are exactly as many paths from $S(f_{c_j})$ to $D(f_{c_j})$ as the number of literals in the clause C_j. For $i = 1, 2, \ldots, n$, there are exactly two paths from $S(f_{u_i})$ to $D(f_{u_i})$. Also files in C never contend with each other for use of a link because their paths to their destinations have no edges in common. The same is true of all the files in U.

In [6] we prove that in the model M_1, *there is a schedule with makespan* $n+2$ *(which is optimal)* if and only if *the corresponding boolean expression has a true assignment.*

Now from the model M_1, we construct $M_2 = (V_2, E_2, S_2, D_2, F_2)$ in which the graph is fully-connected.

1. $V_2 = V_1$
2. $E_2 = \{(a, b) | a \in V_2, b \in V_2, a \neq b\}$
3. S_2, D_2, and F_2 are defined in order to be consistent with the following:
    (a) if $m^1_{ab} = 1$ then $m^2_{ab} = T + 1$
    (b) if $m^1_{ab} = 0$ and $(a, b) \in E_1$ then $m^2_{ab} = T - 1$
    (c) if $m^1_{ab} = 0$ and $(a, b) \notin E_1$ then $m^2_{ab} = T$

Here m^i_{ab} represents the number of files of type $[a, b]$ in the model M_i. We can check easily that $T = n + 2$ is a *lower bound* for the makespan of any schedule in M_2. If we prove the following theorem then we are done.

Theorem 1 *$T_{opt}(M_1) = T$ iff $T_{opt}(M_2) = T$.*

Proof: Proving the *only if* part is easy and we omit the proof here. For the *if* part, let $T_{\text{opt}}(M_2) = T$ and $\mathcal{S}_2$ be an optimal schedule. For each $f \in F_1$ (the file set in M_1), let P_f be set of all paths from its source to its destination in M_1. If we exhibit one path from P_f for each f such that the paths are edge disjoint and all the edges in these paths are used T times in $\mathcal{S}_2$, then we can easily find a schedule $\mathcal{S}_1$ that has a makespan T; we just use these paths to schedule files in M_1. For this, we need the following key lemma.

Lemma 1 *If a file of type $[a, b]$ uses a vertex c in $\mathcal{S}_2$, where $c \notin \{a, b\}$, in its route to its destination, then there exists a path from a to b in M_1, which contains c.*

Proof: Here it is only required to prove that a path from a to b exists in M_2 such that for each edge (x, y) in it, $m^2_{xy} = T - 1$. If such a path does not exist then a *cut* in M_2 containing (say) w edges can be found which separates a from c or c from b and *at least* $wT + 1$ files must cross this cut. The minimum time it takes to move these files is then given by

$$\lceil \frac{wT + 1}{w} \rceil = T + 1,$$

a contradiction since the redistribution can be achieved in time T. Hence the proof. ∎

Corollary 1 *For all* $(x,y) \in E_2 - E_1$*, if there is a file of type* $[x,y]$ *with the route* $(a_0 = x, a_1, a_2, \ldots, a_r = y)$ *in* $\mathcal{S}_2$*, then there exists a directed path* p_{xy} *in* M_1 *that goes through all the vertices* a_j*,* $0 \leq j \leq r$*.*

Corollary 2 *If* (a,b) *is an edge in* M_1 *then all the* $T-1$ *files of type* $[a,b]$ *in* M_2 *must follow the direct link in the schedule* $\mathcal{S}_2$*.*

Next we show that the paths of Corollary 1 are indeed the paths we wish to exhibit for files in F_1. For a directed path p and an edge (a,b), let $A(p,a)$ be the set of all vertices in the path p that precede a (including a) and $B(p,a)$ the set of all vertices that succeed a (including a).

Lemma 2 *Let* f *be a file of type* $[x,y]$ *such that* $(x,y) \in E_2 - E_1$ *and let* $(a_0 = x, a_1, \ldots, a_r = y)$ *be its route in* $\mathcal{S}_2$*. Then in* M_1*, there exists a directed path* $p_{xy} = (W_1, W_2, \ldots, W_r)$*, such that for* $1 \leq k \leq r$*, (i) each* W_k *is a subpath from* a_{k-1} *to* a_k *and (ii) each edge* (a,b) *in* W_k *is used (in* $\mathcal{S}_2$*) by* $T-1$ *files of type* $[a,b]$ *and a file of type* $[c,d]$ *where* $c \in B(W_k, a)$*,* $d \in A(W_k, b)$*, and* $(a,b) \neq (c,d)$*.*

Proof: By Corollary 1, there is a directed path in M_1 satisfying property (i) stated in the lemma. We prove the lemma by induction on the number of edges in each of the subpaths $W_1, W_2, \ldots, W_r$ (with a slight abuse of the notation) of this directed path. Let $1 \leq k \leq r$ be fixed. When $|W_k| = 1$, by Corollary 2 the result obviously follows. When $|W_k| = l \geq 2$, we note that there are T files of type $[a_{k-1}, a_k]$ and since (a_{k-1}, a_k) is used by f there is at least one file of type $[a_{k-1}, a_k]$ that must follow a path with at least one intermediate vertex. By Corollary 1, we find in M_1 a corresponding path $p_{a_{k-1}a_k} = \{W'_1, W'_2, \ldots, W'_h\}$, that satisfies $|W'_i| < l$, $1 \leq i \leq h$. Applying the induction hypothesis the desired result follows. ∎

Using Lemma 2, we find for each file f in F_1, a path p_f from its source to its destination such that all the edges in this path are used by T different files. Now for any two files f_1 and f_2 in F_1, any two paths, one from P_{f_1} and another from P_{f_2}, have at most one edge in common. If the two chosen paths p_{f_1} and p_{f_2} have a common edge (say) (u,v), then by Lemma 2, (u,v) is used by $T-1$ files of type $[u,v]$ and two additional files, a total of $T+1$ files, a contradiction. Hence all the chosen paths are edge-disjoint. This completes proof of the theorem. ∎

Remark : In the previous proof, the optimal schedule of M_2 can be modified such that every file travels at most two edges. To see this, first observe that every file in $F_2 - F_1$ can follow the direct link while every file f in F_1 follows the path p_f found as above. Now, consider a particular $f \in F_1$ and let $p_f = (z_1, z_2, \ldots, z_{l+1})$, where $l \geq 3$. The following procedure alters the route for the file f such that it has at most two edges but without altering the makespan.

1. Move file f through the path (z_1, z_l, z_{l+1}).
2. For j := l downto 3 do the following: Take a file of type $[z_1, z_j]$ and schedule it to move through the path (z_1, z_{j-1}, z_j).

Thus we also show that the problem of determining an optimal schedule under the restriction that every file uses at most two edges is also an intractable problem.

3. Exact Solution

In this section, we present an integer-programming formulation that will aid in finding an optimal schedule for the file redistribution problem with the restriction that a file moves through at most one intermediate node. In this formulation, x_{ikj} indicates the number of files of type $[i,j]$ that uses the link (i,k) and then the link (k,j) in the optimal schedule; x_{ijj} gives the number of files of type $[i,j]$ that use the direct link (i,j). The formulation is as follows:

$$
\begin{array}{rrcll}
 & \text{Minimize} & & T & \\
\text{s.t.} & \sum_{k \notin \{i,j\}} x_{ikj} + x_{ijj} & = & m_{ij}, \text{ for all } i, j \neq i & (1) \\
 & \sum_{k \notin \{i,j\}} (x_{ijk} + x_{kij}) + x_{ijj} & \leq & T, \text{ for all } i, j \neq i & (2) \\
 & \sum_{k \notin \{i,j\}} x_{ijk} & \leq & T-1, \text{ for all } i, j \neq i & (3) \\
 & \sum_{k \notin \{i,j\}} x_{kij} & \leq & T-1, \text{ for all } i, j \neq i & (4) \\
 & x_{ikj} & \in & Z_{\geq 0}, \text{ for all } i, j \neq i, k \neq j & (5)
\end{array}
$$

Constraint (1) states that m_{ij} files that have to be moved from i to j, either go through an intermediate node or go directly. Constraint (2) states that for a makespan of T, no more than T files can use a link. Constraint (3) (or (4)) states that a link is used as the first (or the second) link in the route by no more than $T-1$ files. It is easy to check that an optimal schedule satisfies the constraints of the IP formulation. Now we give an algorithm for constructing an optimal schedule from the optimal solution. The algorithm will also be useful in constructing close to optimal schedules from the approximate solutions that determine for each two-edge route, the number of files of each type that follow that route.

From the optimal solution, we derive for each link e, the quantities a_e, b_e and d_e that indicate the number of files that use e as the first, second and as the direct link respectively. Next we determine for each two-edge route the set of files that use that route; among files of a given type, this can be done arbitrarily as long as they are consistent with the optimal solution. For a file f, let $R_1(f)$ and $R_2(f)$ be the first (or the direct edge) and second edges in the route of f.

Algorithm Schedule :

```
1. Let $\mathcal{S}(e,t) \leftarrow \epsilon$ for all $e \in E$ and $0 \leq t \leq T-1$. For each $e \in E$, let $U_1(e) \leftarrow \{f | R_1(f) = e\}$, $U_2(e) \leftarrow \{f | R_2(f) = e\}$ and $n_e \leftarrow b_e = |U_2(e)|$.
2. For $t \leftarrow 0$ to $T-2$ do steps 3 to 7.
    3. Let $r \leftarrow T-1-t$; $E_2 \leftarrow \{e | n_e = r\}$ and $E_1 \leftarrow \bigcup_{e \in E_2} \{R_1(f) | R_2(f) = e$ and $f \in U_2(e)\}$.
```

4. Construct a bipartite multi-graph $G_b = (X, E_b, Y)$ where $X = \{x_e | e \in E_2\}$, $Y = \{y_e | e \in E_1\}$ and $E_b = \{e_f = (x_{e_2}, y_{e_1}) | f \in U_2(e_2) \text{ and } R_1(f) = e_1\}$.
5. Find a *Matching* ([5]) M of size $|X|$ in G_b.
6. For each $e_f \in M$, do the following: $\mathcal{S}(R_1(f), t) \leftarrow f$; $U_1(R_1(f)) \leftarrow U_1(R_1(f)) - \{f\}$; $U_2(R_2(f)) \leftarrow U_2(R_2(f)) - \{f\}$; $n_{R_2(f)} \leftarrow n_{R_2(f)} - 1$
7. Let $E_1' \leftarrow \bigcup_{e_f \in M} \{R_1(f)\}$. For each $e \in E - E_1'$, if $\mathcal{S}(e, t) = \epsilon$ and $U_1(e)$ is not empty then do the following : first attempt to choose $f \in U_1(e)$ such that f does not use e as the direct link; otherwise choose any f; $\mathcal{S}(e, t) \leftarrow f$; $U_1(e) \leftarrow U_1(e) - \{f\}$]

8.Now $|U_1(e)| \leq 1$ and $U_2(e) = \emptyset$ for all $e \in E$. For each $e \in E$, do the following: if $U_1(e) = \{f\}$, then $\mathcal{S}(e, T-1) \leftarrow f$; Let $t_e \leftarrow \max\{a_e + d_e - 1, 0\}$.

9. For $t \leftarrow 0$ to $T - 1$ do step 10

10. For each $e \in E$, if $\mathcal{S}(e, t) \neq \epsilon$ and e is not the direct link for $f = \mathcal{S}(e, t)$ then do the following : let $e' \leftarrow R_2(f)$; $t_{e'} \leftarrow \max(t, t_{e'}) + 1$; $\mathcal{S}(e', t_{e'}) \leftarrow f$

We first prove that step 2 schedules each link e so that all the files that use them either as direct link or as the first edge in their route, do so before the time unit $a_e + d_e$. This is also done in a manner so that a file that uses a second link e arrives at the intermediate node so as to be able to use it at time $a_e + d_e \leq t \leq T - 1$. Step 6 allows this to be achieved by first scheduling those edges which are needed to transfer files that have to use the second edges immediately upon their arrival to the intermediate nodes. For $0 \leq t \leq T - 1$, let $r^t, E_1^t, E_2^t, U_1^t(e), U_2^t(e)$ and G_b^t be the values of $r, E_1, E_2, U_1(t), U_2(t)$ and G_b at the beginning of iteration t in step 2. For each node y, its indegree and the set of neighbours in G_b^t are given by $\deg^t(y)$ and $\Gamma^t(y)$ respectively.

Lemma 3 *For all* $0 \leq t \leq T - 2$, $deg^t(y) \leq r^t = T - 1 - t$ *for all* $y \in Y^t$ *of* G_b^t.

Proof: For $t = 0$ this is obvious since for $y_e \in Y^0$, $\deg^t(y_e) = a_e \leq T - 1$. For $t \geq 1$, we note that if $\deg^t(y_e) > r^t$ for some y_e then it is obvious that $\mathcal{S}(e, t') = \epsilon$ for some $t' < t$; otherwise constraint (2) of the IP will be violated. But by construction of step 7 of the algorithm a file in $U_1^t(e)$ could have been scheduled at time t' and hence a contradiction. ■

Lemma 4 *Let* (X, E, Y) *be a finite bipartite multi-graph in which the maximal degree of a node is* p. *Let* X_1 *be the set of nodes in* X *with degree equal to* p. *Then there exists a matching* M *which matches* X_1 *to a subset of* Y.

Proof: See [1], pp.46 ■

By lemmas 3 and 4, we see that step 6 is always possible. By the construction of steps 6 and 7 we see that each link e is scheduled to be used at all time units in the interval $[0, a_e + d_e)$ by only those files that use it either as direct or the first edge. Step

9 then schedules the links for use of files that use them as second edges. Here it is only required to prove that for each link e, t_e never exceeds $T-1$. If $t_e = T$ during some iteration, then let $t < T$ be the maximum value for which $\mathcal{S}(e,t) = \epsilon$; this t must exist, otherwise the constraint (2) of the IP will be violated. At time $t-1$, $T-1-(t-1)$ files must be in $U_2(e)$ and hence by step 6, a file should have been scheduled to arrive at the intermediate node so as to use e at time t, a contradiction.

Regarding time complexity, we see that all the steps except step 5 take $O(|E|T)$ time. Using the algorithm of [5], step 5 can be completed in time $O((r^t+1)|X^t|^{3/2})$ time for each iteration $0 \le t \le T-2$. Since $|X^t| \le |E|$, for all t, the total complexity of the algorithm is given by $O(n^3T^2 + n^2T)$. But $T \le m$ where m is the total number of files and hence the complexity of the algorithm is $O(n^3m^2)$.

Remark : We can achieve redistribution of files by first transferring files to a fixed node and then transferring them from this fixed node to their appropriate destinations. Hence by virtue of Theorem 2, the optimal makespan is $O(m/n)$. where n is the number of nodes in the network. If our conjecture that there is always an optimal two-edge schedule were true then the complexity of the scheduling algorithm becomes $O(m^2n)$.

4. Special Case

In this section, we give a polynomial-time algorithm for finding an optimal two-edge schedule in the case when there is *only one* source and k destinations. The case of multiple sources and a single destination is similar and hence we will not discuss it here. Here the optimal makespan depends only on k, the number of nodes n, and the total number of files m. Let $T_{opt}^{k,m}$ represent the minimum makespan in this case.

Lemma 5 $T_{opt}^{1,m} = \lceil \frac{m+n-2}{n-1} \rceil$

Proof: The proof follows from the observation that $T_{opt}^{1,m} = \lceil \min_{1\le x\le m} \max(x, \frac{m-x}{n-2}+1) \rceil$. ∎

Theorem 2 *Let* $T' = \lceil \frac{m}{n-1} \rceil$.

$$T_{opt}^{k,m} = \begin{cases} T'+1 & \textit{if } kT' + (n-k-1)(T'-1) < m \\ T' & \textit{otherwise} \end{cases}$$

Proof: In the first case when $kT' + (n-k-1)(T'-1) < m$, we see that the optimal makespan is greater than T'. Also,

$$\begin{aligned} T_{opt}^{k,m} &\le T_{opt}^{1,m} = \lceil \frac{m+n-2}{n-1} \rceil \\ &\le \lceil \frac{m}{n-1} \rceil + 1 = T' + 1. \end{aligned}$$

In the second case when $kT' + (n-k-1)(T'-1) \ge m$, we see that $T_{opt}^{k,m} \ge T'$, always a trivial lower bound for makespan. By comparing with the case when there are $m-(k-1)$

of files of the same type, and $k-1$ other files, each of a different type, we can deduce $T_{opt}^{k,m} \leq T_{opt}^{1,m-(k-1)}$. Therefore, we have

$$\begin{aligned} T_{opt}^{k,m} &\leq T_{opt}^{1,m-(k-1)} \\ &= \lceil \frac{m-(k-1)+n-2}{n-1} \rceil \\ &\leq \lceil \frac{m-(m-(T'-1)(n-1)-1)+n-2}{n-1} \rceil \\ &= T'. \end{aligned}$$

Hence the proof. ■

Based on Theorem 2, we present the following algorithm to find an optimal two-edge schedule for the single source case. Without loss of generality, assume that node 0 is the source. For $1 \leq i \leq n-1$, let m_i be the number of files of type $[0, i]$. In the schedule, we only determine m_{ij}, the number of files of type $[0, j]$ that go through i; m_{ii} is the number of files that use the direct link $(0, i)$. Using this information, node 0 sends the files to the different nodes with a header for each file specifying the final destination. An intermediate node can send a file to its destination as soon as it receives it without any delay.

Algorithm Single Source Schedule :

```
1. Let $m = \sum_{i=1}^{n-1} m_i$. Determine $T_{opt}^{k,m} = r$ where $k = |\{i | m_i > 0\}|$.

2. Without loss of generality, let $m_i < r$ for $1 \leq i < p$ and $m_i \geq r$ for $p \leq i \leq n-1$.
   For each $1 \leq i \leq p$, do the following: $m_{ii} \leftarrow r$; let $b_i \leftarrow r - 1 - max(0, m_i - 1)$.
   Let $l \leftarrow \min\{i | m_i > 0 \text{ and } m_i < r\}$.

3. For $i \leftarrow p$ to $n-1$ do steps 4-5

4. $m_{ii} \leftarrow r$; let $a \leftarrow m_i - r$;

5. While $a > 0$ repeat the following : $c \leftarrow \min(a, b_l)$; $m_{li} \leftarrow c$; $b_l \leftarrow b_l - c$; $a \leftarrow a - c$;
   if $b_l = 0$ then $l \leftarrow l + 1$.
```

This algorithm has $O(n+m)$ complexity in the worst case. A simpler routing policy for node 0 is to reserve one file each for a destination that needs to be directly transmitted at time $T-1$; at other times, the remaining files are arbitrarily sent to the other nodes along the free links with destination headers. This optimal schedule achieves the same makespan but may tend to increase the average transfer time for a file.

5. Conclusions

In this paper we proved that the problem of determining an optimal schedule for the redistribution of files in a fully-connected network is NP-hard and remains so with the restriction that no file uses more than two edges during its transfer. We gave an IP formulation for finding an optimal schedule with this restriction and an efficient algorithm to construct an optimal schedule from the optimal solution. The IP formulation has the nice property that the dimensions of the constraint matrix depend only on the number of nodes and not on the number of files to be redistributed. But for large networks, approximate solution strategies need to be used and this will be the subject of our future work. We also provided an efficient algorithm for the special case when there is a single source or a single destination. Finally we will attempt to prove or disprove our conjecture that there is always an optimal two-edge schedule.

Acknowledgements : This work has been supported by NSF grant IRI 8716798. We also thank the University of Puerto Rico for its support.

References

[1] R. A. Brualdi, "Transversal Theory and Graphs," *Studies in Graph Theory, Vol.11, Part I*, 1975, pp.23-88.

[2] Hyeong-Ah Choi , "Scheduling Data Transfers in Networks," *Ph. D. thesis, Northwestern University*, 1986.

[3] E. G. Coffman, JR, M. R. Garey, D. S. Johnson and A. S. Lapaugh, "Scheduling File Transfers," *SIAM J. Comput.* , Vol. 14, No. 3, August 1985, pp. 744-780.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman and Company, San Fransisco, 1979.

[5] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Jornal on Computing*, 2 (Dec. 1973),pp. 225-231.

[6] P.I. Rivera, R. Varadarajan and S.B. Navathe, "Scheduling Data Redistribution in Distributed Databases", Technical Report, Database Research and Development Center, University of Florida, 1989.

[7] L. G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM Journal on Computing*, Vol.11, 2, May 1982, pp. 350-361.

OPTIMAL ALGORITHMS FOR ALL TO ALL BROADCAST*

A. BAGCHI[1], S. L. HAKIMI[1], and E. F. SCHMEICHEL[2]

1 Department of Electrical Engineering and Computer Science, University of California, Davis, CA 95616
2 Department of Mathematics and Computer Science, San Jose State University, San Jose, CA 95192. The work of this author is supported in part by the National Science Foundation under Grant DMS-8904520.

ABSTRACT

Consider a network connected with half-duplex links in which each unit initially knows only its own identity and the identity of its immediate neighbors. Suppose each unit wishes to send its unique message to all other units in the network. This is called gossiping. We give a distributed algorithm, which is optimal in the number of transmissions it requires, for gossiping. We give a second algorithm for gossiping in networks which is optimal with respect to the overall utilization of the network. We also show that our algorithm can be used, with slight modifications, in broadcast (radio) networks.

I. INTRODUCTION

Our terminology and notation will be standard. A good reference for any undefined terms is [2].

Consider a computer network of n units represented by a connected, symmetric directed graph $G(V, E)$ with vertex set V and edge set E, where by symmetric directed graph we mean that if $(u,v) \in E$, then $(v,u) \in E$. In particular, the elements of V represent the units (i.e., processors) in the network, and each directed edge (u,v) represents a direct one-way communication link through which communication from u to v can take place. Suppose that a unit has a unique message which must be sent to all other units. This is termed "broadcasting." If each unit wishes to broadcast its message, it is "all to all broadcasting," commonly referred to as "gossiping". The gossiping problem arises in computer networks as it is often necessary for each unit to maintain up-to-date information regarding accounting, resource utilization, processor loads and the lengths of the queues, etc. of all other units. In mobile radio networks, where individual units change their positions, this allows units to maintain routing tables by periodically exchanging information on the list of neighbors of each unit.

Due to its practical importance and the interesting theoretical problems it poses, gossiping has received considerable attention for quite some time. Although the original problem presumed two-way communication between units (the "telephone" model), a number of researchers have looked at communication

---

* Supported by the National Science Foundation under Grant NCR-8716876.

models like ours, in which a transmission can take place in only one direction at a time (referred to as the "telegraph" model). Hedetniemi et al. [4] provide a detailed and comprehensive survey of research on this problem and its variants. However, all the results that we have come across require the existence of an observer with global and complete knowledge of the network. This assumption does not seem realistic in large distributed networks.

In this note, we assume that the network is distributed; that is, each unit initially knows only its own identity and the identity of its immediate neighbors and there is no observer with global knowledge of the network who oversees and controls the scheduling of the transmissions. We also assume that the duration of each transmission is at least as long as the total time needed to transmit all the messages (i.e., if all the messages were available at a node, they could be sent during a single transmission). Finally, we assume that all transmissions are error-free, and that the units cannot alter any message during the algorithm.

We give an algorithm which allows each unit to learn the messages of every other unit using $3(n-1)$ transmissions in total, where n is the number of units in the network. (By comparison, Harary and Schwenk [3] and Golumbic [1] have shown that $2(n-1)$ transmissions are necessary and sufficient when we have an observer with global knowledge of the network, not counting any control messages that the observer might have to send to the units.) We also show that, with slight modifications, our algorithm can be used for gossiping in radio (broadcast) networks, in which a unit can send messages to all of its immediate neighbors using a single transmission (referred to as a "shout" in the gossip literature [4]). Even though radio networks have received considerable attention, we are not aware of any such results.

We also consider gossiping when each of the individual messages is moderately large so that transmitting k messages would require k time units. In this case it becomes important to minimize the number of messages sent on the network. In the optimal case, each unit should receive every message exactly once. We give an optimal distributed algorithm to accomplish this, that does not require any control messages; that is, the only messages sent are the ones actually involved in the gossip.

II. THE ALGORITHM

Our algorithm may be sketched quite simply as follows: Given $v_0 \in V$, we first use $n-1$ transmissions to build the greedy (i.e., lexicographic first) depth-first tree T rooted at v_0, and then $n-1$ additional transmissions for all the units to send their messages up the tree T to v_0. After v_0 has learned all the messages, we use a final $n-1$ transmissions for v_0 to broadcast the total message down the tree T to the other units.

Before giving the algorithm, we introduce some notation. For each $v \in V$, we will use $N(v) = \{w \in V | (v,w) \in E\}$ to denote the set of neighbors of v in G. We will use m_v to denote v's message. Initially each unit v knows only m_v and the identity of the units in $\{v\} \cup N(v)$. At the end of the algorithm, every unit will know $\bigcup_{v \in V} m_v$. Letting v_0 be an arbitrary (starting) vertex in V, we now describe the algorithm.

```
Algorithm Distributed_Gossiping;
begin;
{$V_m$ is the set of units already part of the tree}
{for any unit $v$, $C(v)$ is the set of its children in the tree}
{$M(v)$ is the messages currently known at $v$, initially $M(v) = m_v$ for $v \in V$}
{initialization}
        each $u \in V$, $C(u) = \phi$
        each $u \in V$, $M(u) = m_u$
        $v \leftarrow v_0$;
        $V_m \leftarrow \{v_0\}$;
{forming the depth first tree $T$}
(1)     if $N(v) - V_m \neq \phi$ then
        begin
                choose $w \in N(v) - V_m$;
                $V_m \leftarrow V_m \cup \{w\}$;
                transmit $V_m$ from $v$ to $w$;
                $f(w) \leftarrow v$;  {w sets its father as v};
                $v \leftarrow w$;
                go to (1);
        end; {if}
        else if $N(v) - V_m = \phi$ and $v \neq v_0$ then
        {passing messages up $T$ to $v_0$}
        begin
                $u \leftarrow f(v)$;
                transmit $M(v)$ from $v$ to $u$;
                $M(u) \leftarrow M(u) \cup M(v)$;  {$u$ updates its message}
                $C(u) \leftarrow C(u) \cup \{v\}$;
                $v \leftarrow u$;
                go to (1);
        end; {else if}
```

```
{at this point $M(v_0)$ consists of all messages and needs to be broadcast to all units in $V$}
    else {$N(v) - V_m = \phi$ and $v = v_0$}
    {$V_t$ is the set of units which have $M(v_0)$. Each unit in $V_t$ transmits it to all of
    its children. Once done a unit is removed from $V_t$ and halts.}
    begin
        $V_t \leftarrow \{v_0\}$;
        each $v \in V_t$ do
        begin
            while $C(v) \neq \phi$ do
            begin
                pick $u \in C(v)$;
                $C(v) \leftarrow C(v) - \{u\}$;
                transmit $M(v_0)$ from $v$ to $u$;
                $V_t \leftarrow V_t \cup \{u\}$;
            end;
            $V_t \leftarrow V_t - \{v\}$;
            Halt; {unit $v$ halts}
        end; {each .. do}
    end; {else}
end. {Algorithm Distributed_Gossiping}
```

Proofs of the validity of the algorithm and the fact that it requires at most $3(n-1)$ transmissions are quite simple, and are omitted.

We now show that the above algorithm is very close to being optimal. Consider a "star" network with n units, u_0, ..., u_{n-1}, in which unit u_0 is connected to all other units, while none of the other units have any links between them. Hence, any message sent by unit u_i to u_j has to go through u_0. In our algorithm we have considered gossiping in a distributed network with an arbitrary starting unit. Let u_0 be that starting unit. Note that each of the other $n-1$ units require some message from u_0 before they know that they are participating in gossiping. One requires two transmissions for each unit u_i before u_0 learns m_{u_i}. Thus one needs $2(n-1)$ transmissions before u_0 knows all the messages. However, u_0 has to send the complete set of messages to each of the remaining units, requiring $3(n-1)-1$ transmissions for gossiping.

Our algorithm can also be easily modified to achieve this bound. Simply send the available set of messages instead of V_m while building the tree T. As a result, the last unit u_l added to T and all its ancestors (these are the units in the direct path between u_l and v_0 in T) will know $\bigcup_{v \in V} m_v$. Hence M(v_0) need not be broadcast to them, thus reducing the number of transmissions by at least 1.

III. RADIO NETWORKS

So far we have considered point-to-point networks. We now modify the algorithm so that it can be used in multihop radio networks. These networks have received considerable attention in recent years. Transmission on these networks is over an inherently broadcast communication medium such as radio. A message transmitted by a unit is heard by all units within the line of sight and within the range of the unit. However, two (or more) transmissions received simultaneously at a unit "collide" and are interpreted as noise and neither is received. Also, a unit cannot receive and transmit simultaneously. We represent such a network by a graph $G = (V,E)$, where each vertex $v \in V$ represents a unit and any two units u, v, which can hear each other's transmission are connected by an edge $(u,v) \in E$. Hence, the units of the set $N(v)$, as defined earlier, hear any message transmitted by unit v. Our goal once again is to enable each unit to learn the messages of every other unit, using at most $3(n-1)$ transmissions.

The above algorithm can be used for radio networks with only slight modifications. Due to the nature of the communication medium, a unit, v, can listen to transmissions by any of its neighbors, hence it can locally keep track of the set $N(v) - V_m$. Thus, during the creation of the depth first tree T, there is no need to explicitly compute or transmit V_m. However, when v selects a unit $w \in N(v) - V_m$, its identity needs to be transmitted, so that only one unit is added to T at a time.

In the first stage of the algorithm, which consists of creating the tree T and sending all the messages m_v to v_0, only one unit transmits at any time, so there is no possibility of a collision. However, once v_0 has received all the messages and $M(v_0) = \bigcup_{v \in V} m_v$ has to be broadcast throughout the network, it seems probable that two or more units will transmit at the same time leading to collisions, and so some units might not receive $M(v_0)$. To avoid this, we restrict every non-leaf unit in T to transmit $M(v_0)$ exactly once. As soon as a unit, v, receives $M(v_0)$ from $f(v)$, it transmits the same message and halts. It is easy to see that because T is a depth first tree, when v is receiving from $f(v)$ no other unit in $N(v)$ is transmitting. Thus every unit receives $M(v_0)$ at least once and without collision.

IV. OPTIMAL COST GOSSIPING IN POINT-TO-POINT NETWORKS

In our previous algorithm the objective was to accomplish gossiping with the minimum number of transmissions. This assumes that the messages are short, so that a transmission time is at least as long as the total time needed to transmit all the messages. However, consider the case when the messages are moderately large, and suppose that it costs k time units to send k messages over a link. Hedetniemi et al. [4] suggest this problem for future studies. In this context, we define the cost of using a network based on the notion that it costs one unit to send any single message over a link. Note that for any network with n units, the overall cost of gossip has to be at least $n(n-1)$. (It should be noted that if we had an observer with global knowledge of the network, Harary-Schwenk's algorithm can be modified to obtain an optimal algorithm with overall cost equal to $n(n-1)$ which is also optimal in the number of transmissions, $2(n-1)$.)

We now outline a distributed algorithm which is optimal with regard to the overall cost of gossip and uses $3(n-1)$ transmissions. The interested reader is referred to [5] for a more detailed and formal description. Here we make the same assumptions about the network and use the same notation as in our previous distributed algorithm (described in Section II). In addition we assume that every unit is aware of the number of units, n, in the network. However, unlike our previous algorithm, this algorithm does not require any control messages.

Consider an arbitrary starting unit $v_0 \in V$. This algorithm also forms a depth first spanning tree T rooted at v_0, but uses the process of formation of T also to exchange messages. It requires every unit to go through two stages. In the first stage, the units are involved in the formation of T along with sending and receiving messages. In the second stage, each unit, if it does not have all of the messages, receives a set of messages from its father (which causes it to have all of the messages), then sends the messages to its children which they do not already have.

To begin with, at each node u we maintain a number of lists: $M(u)$, $M(f(u))$, and $\{M_u(v) \mid v \in C(u)\}$, where $M(u)$ represents all messages currently known at u, $M(f(u))$ consists of all messages that u knows its father has received, $C(u)$ is the current set of children of u in T, and for each $v \in C(u)$, $M_u(v)$ is the set of all messages that u knows are known at v. At the start of the algorithm, $M(u) = m_u$, $C(u)$ is empty and $f(u)$ is undefined. Initially v_0, the root of T, is the active unit.

During the formation of the tree, the active unit u selects $v \in N(u)$ which is not already in T, to be its child and sends $M(u)$ to v. At this moment v becomes a part of T, $f(v) \leftarrow$ u, and $C(u) \leftarrow C(u) \cup \{v\}$ and $M(f(v)) \leftarrow M(u)$. Then v becomes active and similarly, it selects a child w and sends to w the messages $M(v) \leftarrow M(u) \cup \{m_v\}$. If all units in $N(v)$ are already in T, v sends the message $M(v) - M(f(v))$ back to $f(v)$ (which is u so $M(f(v)) = M(u)$) and v enters the second stage and u, knowing $M(u)$ and $M(v) - M(f(v))$, constructs $M_u(v)$ $(= M(u) \cup (M(v) - M(f(v)) = M(u) \cup M(v))$

for its child v, and u becomes active. Now, $M(u) \leftarrow M(u) \cup M(v)$ and u proceeds as the active unit. When v enters the second stage it remains idle until activated. The first stage ends when v_0 is active and v_0 has no child which is not already in T.

In the second stage of the algorithm, each unit that has received all of the messages becomes active. (Note that v_0 will be an active unit.) Let u be an active unit. For each $v \in C(u)$, u sends the message $M(u) - M_u(v)$ to vertex v and v is activated. Unit v updates $M(v) \leftarrow M(v) \cup (M(u) - M_u(v))$. Once u sends the messages to all of its children, u ceases to be active. Each activated vertex proceeds to do exactly as above in parallel and independently of all other active units. The algorithm ends when there are no active vertices.

Once the basic idea of the algorithm is understood, its validity becomes self-explanatory. Furthermore, it is easy to see that each link in T has exactly n messages sent on it. Thus, the total cost of transmission is $n(n-1)$, which is optimal. It should also be noted that we use at most $3(n-1)$ transmissions. Thus, the algorithm uses the same number of transmissions as the previous algorithm; however, this algorithm requires each unit to know the number, n, of units in the system, which was not required by the previous algorithm.

Finally, it is possible to modify the above algorithm such that the units, u, need not keep track of $M_u(v)$ for all $v \in C(u)$. The resulting algorithm will be optimal in the above sense but may require more than $3(n-1)$ transmissions [5].

V. CONCLUSION

We have demonstrated that we can complete the task of transmitting every unit's message to every other unit using $3(n-1)$ transmissions in a distributed network. We have also shown that our algorithm is optimal for point-to-point networks. We have not been able to show that there are instances of the problem when $3(n-1)$ transmissions are required in radio networks. Thus, the question of optimality for radio networks remains open.

In our algorithm we have assumed the existence of a unique but arbitrary unit, v_0, which starts the algorithm. It might be interesting to develop algorithms for networks without a "leader" so that more than one unit might start the algorithm. Another interesting problem might be to develop algorithms which make optimal use of transmissions in parallel.

REFERENCES

[1] M.C. Golumbic, “The general gossip problem,” IBM Research Report 4977, 1974.

[2] F. Harary, Graph Theory, Addison-Wesley, Reading, MA, 1969.

[3] F. Harary and A.J. Schwenk, “The communication problem on graphs and digraphs,” J. Franklin Institute, 297(1974), pp. 491-495.

[4] S.M. Hedetniemi, S.T. Hedetniemi and A.L. Liestman, “A survey of gossiping and broadcasting in communication networks,” Networks, vol. 18, no. 4, Winter 1988, pp. 319-349.

[5] A. Bagchi, Ph.D. Dissertation (in preparation), Department of Electrical Engineering and Computer Science, University of California, Davis, CA 95616.

EDGE-LABELLED GOSSIP IN ROUNDS

ARTHUR L. LIESTMAN
School of Computing Science, Simon Fraser University, Burnaby, British Columbia V5A 1S6, Canada

DANA RICHARDS
Department of Computer Science, University of Virginia, Charlottesville, Virginia 22901, U.S.A.

ABSTRACT

In this paper, we consider a gossiping problem in which individuals in a network exchange information periodically according to a fixed schedule. In particular, a proper edge-coloring can be considered as a schedule for the calls of a gossiping scheme. We have investigated the time required to complete gossiping in graphs under various colorings. In particular, we have determined the minimum time to complete gossiping under any proper edge-coloring of a path and given bounds on the time required to complete gossiping in cycles and in trees with bounded degree.

DEFINITIONS

Gossiping is the process in which each member of a set of individuals knows a unique piece of information initially and must learn all of the information of the group through some set of communications (calls). This process is modeled as a problem on graphs in which the vertices represent individuals and edges represent allowed communication links. Several variations of this problem have been considered (see [2]). We are interested in a particular situation in which specific calls are made periodically according to a predetermined schedule. In practical terms, such a schedule allows efficient communication while avoiding the possibility of message collisions or congestion.

Let G be any connected graph on n vertices and let C be a proper k edge-coloring of G which assigns the colors $1, 2, \ldots, k$ to the edges of G such that no vertex is incident on more than one edge of any color i. (Henceforth, all edge-colorings will be assumed to be proper and we will omit the word.) If all k colors are used in the edge-coloring, we say that it is a **strict k edge-coloring**. At time 0, every vertex has a unique piece of information known only to itself. At each time t, $t = 1, 2, 3, \ldots$, each pair of vertices which are joined by an edge colored $i \equiv t (mod\, k)$ exchange all of the information they know at that time. Note that operations on colors are performed modulo k, adopting the convention that color k is used in place of color 0. Let t_C denote the earliest time t at which every vertex has learned all n pieces of information given the edge-coloring C. We say that G **completes gossiping** under edge-coloring C at time t_C. The k **gossip time** of G, denoted $\overline{g}_k(G)$, is defined to be the minimum t_C over all k edge-colorings C of G. The **strict k gossip time** of G, denoted $g_k(G)$, is defined to be the minimum t_C over all strict k edge-colorings C of G. It follows from the definitions that $g_k(G) \geq \overline{g}_k(G)$.

GOSSIPING ON PATHS

Let $P_n = (v_0, v_1, \ldots, v_n)$ denote the path consisting of n edges and $n+1$ vertices. It is convenient to use an n character k-ary string $c_1 c_2 \ldots c_n$, where $c_i \in \{1, 2, \ldots, k\}$ for all i, to denote a particular k edge-coloring of P_n. In particular, a (proper) k edge-coloring corresponds to such a string in which $c_i \neq c_{i+1}$ for $1 \leq i < n$. A strict k edge-coloring has the additional property that for each $j \in \{1, \ldots, k\}$ there is at least one i, $1 \leq i < n$, such that $c_i = j$.

The following simple observations, stated without proof, will be useful below.

1. For a given k edge-coloring C of P_n, let t_{0n} denote the time required for a message to travel from v_0 to v_n and let t_{n0} denote the time required for a message to travel from v_n to v_0. The k gossip time of P_n under edge-coloring C is the maximum of these two times, that is, $t_C = \max\{t_{0n}, t_{n0}\}$.
2. The k edge-coloring $c_1 c_2 \ldots c_n$ and its reverse, $c_n c_{n-1} \ldots c_1$, complete gossiping at the same time.

Lemma 1: Given any k edge-coloring C of P_n with $c_1 > 1$, there is another k edge-coloring C' of P_n such that $c'_1 = 1$ and $t_{C'} \leq t_C$.

Proof: Since the k edge-coloring C and its reverse C' complete gossiping at the same time, if $c_n = 1$ then $C' = c_n c_{n-1} \ldots c_1$ is such a k edge-coloring.

We can now restrict our attention to k edge-colorings C of P_n with $c_1 > 1$ and $c_n > 1$. Without loss of generality, we can assume that $c_1 < c_n$. From such a C, we construct a new k edge-coloring $C' = c'_1 c'_2 \ldots c'_n$ by shifting the numbers corresponding to the colors by c_1 modulo k. For each i, let $c'_i = (c_i - c_1 + 1) \pmod{k}$, recalling that 0 is represented by color k. In particular, this means that $c'_1 = 1$. Messages traveling through any interior node of P_n encounter the same delays with either coloring since the difference between two consecutive labels remains the same modulo k. Furthermore, the endpoints send their own messages $c'_1 - 1$ time units earlier than under the previous edge-coloring. Thus, in this case $t_{C'} \leq t_C$. □

Note that the edge-coloring constructed in the proof of Lemma 1 is strict if and only if the initial edge-coloring is strict.

Corollary 2: There is a strict k edge-coloring C of P_n with $c_1 = 1$ which completes gossip in time $g_k(P_n)$ and a k edge-coloring C' of P_n with $c'_1 = 1$ which completes gossip in time $\overline{g}_k(P_n)$.

Proof: This follows immediately from Lemma 1. □

Thus, in searching for (strict) k edge-colorings of paths that complete gossiping in the minimum time, we need only consider those in which the first edge is colored 1.

Theorem 3: $$g_k(P_n) \geq \overline{g}_k(P_n) \geq \begin{cases} \dfrac{k(n-1)}{2} + 1, & n \text{ odd}, n \geq k \geq 2 \\ \dfrac{kn}{2} + 1, & n \text{ even}, n \geq k \geq 2. \end{cases}$$

Proof: The first inequality follows from the definitions of $g_k(G)$ and $\overline{g}_k(G)$. Note that the requirement that $n \geq k$ is due to the definition of strict k edge-coloring. The same bound actually holds for $\bar{g}_k(P_n)$ for any $n \geq 2$.

We begin by considering the simple case $k=2$. Clearly, $\bar{g}_2(P_n) \geq n$ (and hence $g_2(P_n) \geq n$) since the information can travel over at most one edge per time unit and the two endpoints v_0 and v_n are at distance n from each other. For even n, the only 2 edge-coloring with $c_1 = 1$, 12121...212, is strict and has $c_n = 2$. The information from v_n leaves v_n at time 2 and is thereby delayed by one time unit. Thus, $\bar{g}_2(P_n) \geq n+1$ (and $g_2(P_n) \geq n+1$) for even n. (In fact, we have shown equality for even n.)

Next, we consider $k=3$, that is, we show that $\bar{g}_3(P_n) \geq \frac{3n-1}{2}$ for odd $n \geq 2$ and that $\bar{g}_3(P_n) \geq \frac{3n}{2}+1$ for even $n \geq 2$. We will then generalize this argument for larger k.

In any 3 edge-coloring of P_n with adjacent edges (u, v) and (v, w) colored i and j, respectively, we can assume that $j = i+1$, without loss of generality. Any information traveling in the direction from u to w is forwarded by v immediately so that the information is known to w exactly 2 time units after it is sent by u. However, any information traveling from w to u is delayed for one time unit at v and it is known to u exactly 3 time units after it is sent by w. Thus, any intermediate vertex in the path P_n adds one unit of delay to a message traveling through it in one direction and no delay to a message traveling through it in the opposite direction.

In order to complete gossiping in a given edge-coloring of P_n, the two endpoints must each learn each other's information. In particular, this means that one message must travel from v_0 to v_n and another must travel from v_n to v_0. The time required for either of these messages to arrive at is destination is a lower bound on the time required to complete gossiping. Each of these messages passes over n edges and encounters $n-1$ intermediate vertices. Each message requires exactly one unit of time to travel over each edge and each intermediate vertex delays exactly one of the messages by one time unit. Thus, the sum of the times taken by these two messages is $3n-1$ and $\bar{g}_3(P_n) \geq \lceil \frac{3n-1}{2} \rceil$.

We can improve this lower bound slightly for the case of even n. In particular, let $n = 2p$ while $k=3$. Let us assume, by way of contradiction, that gossip can be completed in P_n under some 3 edge-coloring C in time $\lceil \frac{3n-1}{2} \rceil = 3p$. As above we will restrict our attention to the messages sent between the endpoints. Since we can also assume that $c_1 = 1$, the endpoint v_0 must learn v_n's information at some time $t \equiv 1 \pmod 3$. Since $t \leq 3p$, this must occur no later than at time $3p-2 \equiv 1 \pmod 3$. Thus, the message from v_n to v_0 encountered at most $3p-2-n = p-2$ delays at intermediate vertices. Since the total number of delays incurred by the two messages sent between the endpoints is $n-1$, the message from v_0 to v_n must encounter at least $(n-1)-(p-2) = p+1$ delays and can arrive at v_n no sooner than time $n+p+1 = 3p+1$, contradicting the assumption.

This argument is easily generalized for arbitrary k. In any k edge-coloring of P_n with adjacent edges (u, v) and (v, w) colored i and j, respectively, we can assume that $j > i$, without loss of generality. Any information traveling from u to w is delayed $j-i-1$ time units at vertex v whereas any information traveling from w to u is delayed $k-(j-i)-1$ time units at vertex v. Each of the messages traveling between v_0 and v_n passes over n edges and encounters $n-1$ intermediate vertices. Each message requires exactly one unit of time to travel over each edge while each intermediate vertex introduces a total of $k-2$ units of delay to the two messages. Thus, the sum of the times taken by these two messages is $2n+(n-1)(k-2)=(n-1)k+2$ and $\overline{g}_k(P_n) \geq \lceil \frac{(n-1)k+2}{2} \rceil$.

As before, we can improve this bound slightly for the case of even n. In particular, let $n=2p$. We assume, by way of contradiction, that gossip can be completed in P_n under some k edge-coloring C in time $\frac{n}{2}k$. Since $c_1=1$, v_0 must learn v_n's information at some time $t \leq \frac{n}{2}k=pk$ where $t \equiv 1 \pmod{k}$. So, in fact, v_0 must learn v_n's information at some time $t \leq (p-1)k+1$. Since the path contains n edges, each of which requires one unit of time to cross, the message from v_n to v_0 incurred at most $(p-1)k+1-n=p(k-2)-k+1$ delays at the intermediate vertices. Since a total of $(n-1)(k-2)$ delays are incurred by the two messages sent between the endpoints, the message from v_0 to v_n must encounter at least $(n-1)(k-2)-(p(k-2)-k+1)=p(k-2)+1$ delays and can arrive at v_n no sooner than time $p(k-2)+1+2p=pk+1$. □

Before stating general upper bounds for $g_k(P_n)$ and $\overline{g}_k(P_n)$, we consider the special case $n=k$ under strict edge-coloring.

Lemma 4: $g_k(P_k) = \begin{cases} \frac{k(k-1)}{2}+2, & \text{odd } k \geq 3 \\ \frac{k^2}{2}+1, & \text{even } k \geq 4. \end{cases}$

Proof: For odd $k \geq 3$, the strict edge-coloring $135\ldots k(k-1)(k-3)\ldots 2$ has $t_{0n}=\frac{k(k-1)}{2}+2$ and $t_{n0}=\frac{k(k-1)}{2}+1$. In this case, the lower bound from Theorem 3 can be improved. The edge-coloring must be a permutation of $1,2,\ldots,k$. In particular, if we assume that the edge from v_1 to v_2 is colored 1, then the edge from v_k to v_{k-1} must be colored i for some $i>1$. This means that a delay of $i-1 \geq 1$ time units is introduced at v_k. The sum of the delays incurred by the two messages sent between the endpoints is at least $k(k-1)+1$. Since every edge must be traversed by each of these two messages, the sum of the times taken by the two messages is $2k+(k-2)(k-1)+1$, so $g_k(P_k) \geq \lceil \frac{2k+(k-2)(k-1)+1}{2} \rceil = \frac{k(k-1)}{2}+2$.

For even $k \geq 4$, the strict edge-coloring $135\ldots(k-1)k(k-2)(k-4)\ldots 2$ has $t_{0n}=\frac{k^2}{2}-k+2$ and $t_{n0}=\frac{k^2}{2}+1$. The lower bound comes directly from Theorem 3. □

Theorem 5: $\overline{g}_k(P_n) = \begin{cases} \frac{k(n-1)}{2}+1, & n \text{ odd}, n \geq k \geq 2 \\ \frac{kn}{2}+1, & n \text{ even}, n \geq k \geq 2. \end{cases}$

Proof: For odd n, the non-strict 2 edge-coloring $12121\ldots 21$ has $t_{0n}=t_{n0}=\frac{k(n-1)}{2}+1$. For even n, the non-strict edge-coloring $1212\ldots 12$ has $t_{0n}=\frac{kn}{2}$ and $t_{n0}=\frac{kn}{2}+1$. The matching lower bounds come directly from Theorem 3. □

Theorem 6: $g_k(P_n) = \begin{cases} \frac{k(n-1)}{2}+1, & n \text{ odd}, n > k \geq 2 \\ \frac{kn}{2}+1, & n \text{ even}, n \geq k \geq 2. \end{cases}$

Proof: We will prove the following claim from which the result follows. "There exist colorings with the following properties: For even n, $c_n = 2$, $t_{0n} = \frac{kn}{2} - k + 2$ and $t_{n0} = \frac{kn}{2} + 1$. For odd n, $c_n = 1$ and $t_{0n} = t_{n0} = \frac{k(n-1)}{2} + 1$.

We prove this claim by induction. Regardless of the parity of k, the basis of the induction is for even n. When k is even, the basis is given in the proof of Lemma 4. When k is odd, the sequence $123\ldots(\frac{k+1}{2})k(k-1)\ldots(\frac{k+1}{2}+1)2$ with $n = k+1$ provides the basis.

Note that if a color sequence ends with a 2, then appending a 1 causes the leftward time (t_{0n}) to increase by $k-1$ while the rightward time (t_{n0}) stays the same. Further, if a color sequence ends with a 1, then appending a 2 causes the leftward time to increase by 1 and the rightward time to increase by k.

Using these observations it is easy to show that if the claim is true for an even (odd) n then it follows for the next odd (even) $n+1$. In the even case a 2 is appended and in the odd case a 1 is appended. For example, suppose the leftward time for an odd n is $\frac{k(n-1)}{2}+1$ and the final color is 1. If a 2 is appended, the leftward time for the even $n+1$ is $(\frac{k(n-1)}{2}+1)+1 = \frac{k(n+1)}{2} - k + 2$.

The lower bounds come directly from Theorem 3. □

Lemma 7: Given any (strict) k edge-coloring C of P_n, $t_C \leq k + (k-1)(n-1)$.

Proof: We construct a specific coloring strict k edge-coloring C of P_n which maximizes the time required for a message to travel from v_0 to v_n. In order to delay the starting time of the message from v_0 as long as possible, let $c_1 = k$. For any intermediate vertex v_i, the message can be delayed by at most $k-1$ time units. By setting c_{i+1} to $c_i + (k-1) \pmod{k}$ for each i, the message is delayed as much as possible and arrives at v_n at time $k + (k-1)(n-1)$. Thus, $t_C = k + (k-1)(n-1)$. It is clear from the construction that no other k edge-coloring of P_n could require more time to complete gossiping. □

GOSSIPING IN TREES OF BOUNDED DEGREE

Let T denote a tree on n vertices. We use d_T to denote the diameter of T and Δ_T to denote the maximum degree of any vertex in T. The subscripts may be omitted when the meaning is clear from context. Clearly, any proper edge-coloring of T must use at least Δ_T colors.

Lemma 8: Given a tree T of diameter d and maximum degree Δ, $g_k(T) \geq \overline{g}_k(T) \geq g_k(P_d)$ for any $k \geq \Delta$.

Proof: The first inequality follows from the definitions of g_k and $\overline{g}_k$. The second follows from the observation that information must travel between two vertices which are at a distance Δ apart and the fact that between any such pair of vertices there is a unique path in T. □

Let us consider the special case of trees T where $\Delta_T = 3$. For a fixed diameter d, consider the largest (most vertices) tree with diameter d and maximum degree 3 which we will call T_d. T_d has $1 + 3\sum_{r=1}^{d/2} 2^{r-1}$ vertices if d is even and $2\sum_{r=1}^{(d+1)/2} 2^{r-1}$ vertices if d is odd (see [1]). We can draw T_d

(as an example, T_7 is shown in Figure 1) as two full regular binary trees T_a and T_b rooted at adjacent vertices a and b, respectively, where the height of T_a is $\lceil \frac{d}{2} \rceil$ and the height of T_b is $\lfloor \frac{d}{2} \rfloor$.

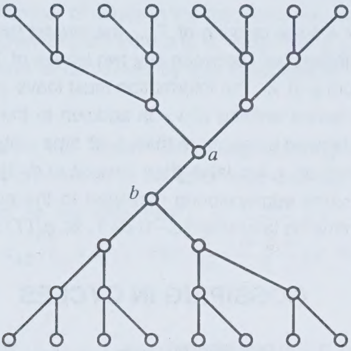


Figure 1: largest tree with diameter 7 and maximum degree 3

Since every tree T with diameter d and maximum degree 3 is a subtree of T_d, any proper edge-coloring of T_d (when restricted to the edges of T) is a proper edge-coloring of T. Thus, any upper bound on the gossip time of T_d is an upper bound on the gossip time of T.

Theorem 9: If T is a tree with diameter d and maximum degree 3, then $\overline{g}_3(T) \leq g_3(T) \leq 2d+1$.

Proof: Consider any proper 3 edge-coloring of T_d. The length of the longest path between any two leaves of T_d, say u and v, is d. Consider the edge labels on the path from u to v. The information must leave u no later than at time 3 and then travel over $d-1$ other edges before arriving at v. In addition to the time taken to travel over these edges, the message may be delayed by no more than 1 time unit at any of the $d-1$ intermediate vertices. Thus, it must arrive at v no later than time $3+(d-1)+(d-1)=2d+1$, so $g_3(T_d) \leq 2d+1$. The same edge-coloring restricted to the edges of T yields a coloring of T which completes gossiping at time no later than $2d+1$, so $g_3(T) \leq 2d+1$. □

An improved lower bound can been shown for the tree T_d.

Theorem 10: $2d-1 \leq \overline{g}_3(T_d) \leq g_3(T_d)$.

Proof: Consider all of the messages travelling from a bottom leaf to a top leaf of the tree T_d (where bottom and top are as depicted in Figure 1). Let c be an internal node in the lower subtree rooted at b. All messages coming up to c either from its left subtree or from its right subtree must be delayed for one unit at c. Let d be an internal node in the upper subtree rooted at a. Messages coming up to any such node d and bound for leaves in the upper subtree rooted at d may be delayed at d. In particular, all such messages destined for nodes in either the left upper subtree or the right upper subtree rooted at d will be delayed for one unit at node d. Hence, at least one message from a bottom leaf to a top leaf will encounter delays at every intermediate step. The total time taken by this message (if there is no initial delay) is $d+(d-1)=2d-1$. □

The proof of Theorem 9 is easily generalized to produce an upper bound on the gossip time in trees with diameter d and maximum degree Δ.

Theorem 11: If T is a tree of diameter d and maximum degree Δ, then $\overline{g}_\Delta(T) \le g_\Delta(T) \le (\Delta-1)d+1$.

Proof: Consider any proper Δ edge-coloring of $T_{\Delta,d}$, the largest tree with diameter d and maximum degree Δ. The length of the longest path between any two leaves of $T_{\Delta,d}$, say u and v, is d. Consider the edge labels on the path from u to v. The information must leave u no later than at time Δ and then travel over $d-1$ other edges before arriving at v. In addition to the time taken to travel over these edges, the message may be delayed by no more than $\Delta-2$ time units at any of the $d-1$ intermediate vertices. Thus, it must arrive at v no later than time $\Delta+(d-1)+(\Delta-2)(d-1) = (\Delta-1)d+1$, so $g_\Delta(T_{\Delta,d}) \le (\Delta-1)d+1$. The same edge-coloring restricted to the edges of T yields a coloring of T which completes gossiping at time no later than $(\Delta-1)d+1$, so $g_3(T) \le (\Delta-1)d+1$. □

GOSSIPING IN CYCLES

Let C_n denote the cycle on n vertices labelled $v_1, v_2, \ldots, v_n$. We can use an n character k-ary string $c_1 c_2 \ldots c_n$ to denote a particular k edge-coloring of C_n. In this case, notice that for the edge-coloring to be proper, it is necessary that $c_1 \ne c_n$. As with paths, any edge-coloring and its reverse complete gossiping at the same time. Furthermore, any edge-coloring $c_1 c_2 \ldots c_n$ and its circular shifts $c_2 \ldots c_n c_1$, $c_3 \ldots c_n c_1 c_2$, ... and $c_n c_1 \ldots c_{n-1}$ all complete gossiping at the same time.

Lemma 12: Given any k edge-coloring C of C_n with $c_1 > 1$, there is another k edge-coloring C' of C_n such that $c'_1 = 1$ and $t_{C'} \le t_C$.

Proof: Since the k edge-coloring C and its circular shifts all complete gossiping at the same time, if $c_i = 1$ then $C' = c_i c_{i+1} \ldots c_n c_1 \ldots c_{i-1}$ is such a k edge-coloring.

We can now restrict our attention to k edge-colorings C of P_n with $c_i > 1$ for all $1 \le i \le n$. Choose i such that $c_i \le c_j$ for all $j \ne i$. Consider the edge-coloring $C' = 1\, c_{i+1} \ldots c_n c_1 \ldots c_{i-1}$, that is, the k edge-coloring obtained from C by shifting c_i into the first position and replacing it with 1. It is easy to see that the time to complete gossip in C' is no more than that in C. By our choice of i, we know that the neighboring edges had larger labels in C. Any message arrives at an endpoint of the relabelled edge at the same time in C' as in C. The message is transmitted over the relabelled edge exactly c_i-1 time units earlier than in C. The message is then forwarded from the other endpoint at the same time as in C. Thus, $t_{C'} \le t_C$. □

Note that if the initial edge-coloring is strict, then the edge-coloring constructed in the proof of Lemma 12 is a circular shift of the original and thus also strict.

Corollary 13: There is a strict k edge-coloring C of C_n with $c_1 = 1$ which completes gossip in time $g_k(C_n)$ and a k edge-coloring C' of C_n with $c'_1 = 1$ which completes gossip in time $\overline{g}_k(C_n)$.

Proof: This follows immediately from Lemma 12. □

Thus, in searching for (strict) k edge-colorings of cycles that complete gossiping in the minimum time, we need only consider those in which the first edge is colored 1.

Theorem 14: $\overline{g}_k(C_n) \geq \frac{(k-1)n}{k} + \frac{4}{k} - 2$ and $g_k(C_n) \geq \frac{(k-1)n}{k} + \frac{4}{k} - 2 + \frac{(k-1)(k-2)}{kn}$.

Proof: Under any edge-coloring C, the information originating at vertex a must arrive at distinct and adjacent vertices b (following a path counterclockwise from a) and c (following a path clockwise from a) in time $\leq t_C$. Let l_{ab} be the number of edges in the (counterclockwise) path from a to b and l_{ac} be the number of edges in the (clockwise) path from a to c. Note that $l_{ab}+l_{ac}=n-1$. Similarly, let d_{ab} and d_{ac} be the number of delays encountered by the messages on these paths. It follows that $t_C \geq l_{ab}+d_{ab}$ and $t_C \geq l_{ac}+d_{ac}$.

Suppose the Theorem is false, i.e., under some edge-coloring C gossip completes in time $t_C < \frac{(k-1)n}{k} + x$, where x depends on whether or not strict edge-coloring is required. Then, for every a, $l_{ab} > \frac{n}{k} - x - 1$, otherwise it must be the case that $l_{ac} \geq \frac{(k-1)n}{k} + x$. Similarly, we can argue that $l_{ac} > \frac{n}{k} - x - 1$. If $S = \Sigma_a\, (l_{ab}+d_{ab}+l_{ac}+d_{ac})$ then $\frac{S}{2n} < \frac{(k-1)n}{k} + x$ must also hold. We show that $\frac{S}{2n} \geq \frac{(k-1)n}{k} + x$.

Let $S = S_l + S_d$, where $S_l = \Sigma_a\, (l_{ab}+l_{ac}) = n(n-1)$ and $S_d = \Sigma_a\, (d_{ab}+d_{ac})$. Consider a fixed vertex p as a varies. Recall that at any p, the sum of the delay incurred by a message traveling clockwise and the delay incurred by a message traveling counterclockwise is $k-2$. For $\frac{n}{k} - x - 2$ choices of a the counterclockwise delay of p must contribute to d_{ab} and for $\frac{n}{k} - x - 2$ choices of a the clockwise delay of p must contribute to d_{ac}. Hence $S_d \geq (k-2)\,n(\frac{n}{k} - x - 2) + S_d'$ where S_d' is the sum of the delays encountered at the beginning for each a that are not counted in the above argument.

For the non-strict case, we know that at each a one of the incident edges is > 1, so there is a delay of at least 1 in that direction. Hence, $S_d' \geq n$. For the strict case, we know that there are choices of a which must have incident edges labelled with 2, 3, ..., k. It follows that $S_d' \geq (n-k) + k(k-1)$.

Using the above inequalities it follow that, in both cases, $\frac{S}{2n} \geq \frac{(k-1)n}{k} + x$. □

Lemma 15: For even $n \geq 4$, $g_2(C_n) = \overline{g}_2(C_n) = \frac{n}{2}$.

Proof: Since the diameter of C_n is $\frac{n}{2}$, we know that $g_2(C_n) \geq \frac{n}{2}$ and $\overline{g}_2(C_n) \geq \frac{n}{2}$ from Theorem 14. The strict 2 edge-coloring 1212...12 completes gossiping in $\frac{n}{2}$ time units. □

Lemma 16: For $n \geq 4$, $g_3(C_n) \leq \lfloor \frac{3n+1}{4} \rfloor$.

Proof: The edge-colorings 1212...123 and 1212...1213 for odd and even n, respectively, complete gossip in $\lfloor \frac{3n+1}{4} \rfloor$ time units. □

Lemma 17: For $n \geq 4$, $\overline{g}_3(C_n) \leq \lceil \frac{n}{2} \rceil + \lceil \frac{n-4}{4} \rceil$.

Proof: The edge-colorings 1212...123 and 1212...12 for odd and even n, respectively, complete gossip in $\lceil \frac{n}{2} \rceil + \lceil \frac{n-4}{4} \rceil$ time units. □

Theorem 18: For $n = ik + j$ such that $1 \le j \le k$, $\bar{g}_k(C_n) \le g_k(C_n) < \frac{(k-1)n}{k} + \frac{(2k-j-1)(k-1)}{k}$.

Proof: For $n = ik+j$ where $2 \le j \le k$, consider the strict edge-coloring $123\ldots k\ldots 123\ldots k123\ldots j$, that is, c_i is set to be $i \bmod k$. Under this particular edge-coloring, information travels much more quickly in one direction (say clockwise) than in the opposite direction (counter-clockwise). The information beginning at any vertex must be received by all the other vertices before gossip is completed. The last vertices to send their information are those vertices v_i where $i \equiv 0 \bmod k$, that is, those vertices incident on edges with labels $k-1$ and k. Such a vertex, say u, begins by sending its information counter-clockwise at time $k-1$ and then sends its message clockwise at time k. The number of vertices informed by time t along a counter-clockwise path from u is roughly $\lfloor \frac{t}{k-1} \rfloor$ for $t \ge k-1$. When proceeding counter-clockwise from v_1, $c_1 = 1$ is followed by $c_n = j$. For some starting vertices, this pair of edges will be traversed counter-clockwise during the gossiping scheme, informing vertices more quickly. Thus, the number of vertices informed by time t along a counter-clockwise path from u is $\ge \lfloor \frac{t}{k-1} \rfloor$ for $t \ge k-1$. In the clockwise direction, $c_n = j$ is followed by $c_1 = 1$, so a message travelling over this pair of edges is delayed by $k-j$ time units at v_1. The number of vertices informed by time t along a clockwise path from u is $\ge t-(k-1)-(k-j)$ for $t \ge k-1$. Thus, all n vertices will know u's information and gossip will be completed at time no later than t such that $n = 1 + (t-(k-1)-(k-j)) + \lfloor \frac{t}{k-1} \rfloor$. Solving for t we get, $t < \frac{(k-1)n}{k} + \frac{(2k-j-1)(k-1)}{k}$.

When $n = ik+1$, the above edge-coloring is not proper. In this case, we consider the strict edge-coloring $123\ldots k\ldots 123\ldots k2$, that is, c_i is set to be $i \bmod k$ for $i < n$ and $c_n = 2$ and can show that $t < \frac{(k-1)n}{k} + \frac{2(k-1)(k-1)}{k}$. □

REMARKS

The following simple bounds on the gossip time of an arbitrary connected graph on n vertices follow from the definition of diameter and Lemma 7.

Lemma 19: Given any n vertex connected graph G with diameter d, $d \le \bar{g}_k(G) \le k+(k-1)(d-1)$ and $d \le g_k(G) \le k+(k-1)(d-1)$.

Except for these trivial bounds, the problem is open for arbitrary connected graphs.

Similar bounds for arbitrary trees may be derived from the definition of diameter and Theorem 11. Otherwise, the problem is open for arbitrary trees.

We believe that the upper bounds of Lemmas 16 and 17 are, in fact, equalities. In particular, equality holds for $4 \le n \le 16$.

REFERENCES

[1] Bollobas, B.
Extremal Graph Theory.
Academic Press, New York, 1978.
page 172.

[2] Hedetniemi, S. T., Hedetniemi, S. M. and Liestman, A. L.
A Survey of Broadcasting and Gossiping in Communication Networks.
Networks 18:319-349, 1988.

TIME-SPACE TRADEOFFS IN NONBLOCKING PARALLEL COMMUNICATIONS[†]

David M. KOPPELMAN[*]
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

A. YAVUZ ORUÇ
Electrical Engineering Department
and
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

ABSTRACT

In this paper, we examine the complexity of nonblocking switching networks with an emphasis on the relations between their costs and path–lengths. Much of the earlier work on the complexity of nonblocking networks deals with deriving bounds on the number of crosspoints in such networks. While such complexity bounds provide valuable information about nonblocking networks, the cost alone is not sufficient to fully characterize their performance. To better evaluate the performance of these networks this paper investigates the basic relations between their costs and path–lengths. It is shown that the cost, C_n, of an n-input nonblocking network for a given path–length, S_n, satisfies the inequality $C_n \geq S_n n^{1+1/S_n}$. It is also shown that, for a given cost, C_n, and fanout, f, the path–length of a nonblocking network satisfies the inequality $S_n \geq n\, log_f^2\, n/(fC_n)$. It is further established that there exist rearrangeable Clos networks which are near optimal in the sense of these bounds. Finally, it is shown that no nonblocking network which is derived from a 3-stage Clos construction is optimal when compared to these bounds.

1. Introduction

Given two finite sets of terminals in space, a key problem which arises in communication theory is to devise a connection structure between these two sets of terminals so as to guarantee the existence of a path between any pair of terminals (one terminal from each set), regardless of all such paths which may already exist between other pairs of terminals. These connection structures are commonly referred to as *nonblocking* networks, and are typically formed by cascading smaller nonblocking networks together in some particular way [3,7]. The smaller nonblocking networks might be implemented using a fundamental connecting device called a *crossbar switch,* (sometimes referred to as a *switch*).

[†] This work is supported in part by the National Science Foundation under Grant No: CCR-8708864.

[*] David M. Koppelman is no longer at RPI. Correspondence can be sent to him at 2735 East 65th St., Brooklyn, NY 11234.

In this paper we shall examine the basic limits and tradeoffs in designing nonblocking networks with a particular emphasis on the inherent relations between their cost and path–length. The cost of a network will be assessed by the number of *programmable edges,* i.e., the edges which connect the inputs and outputs of its switches. Thus, a crossbar switch with m inputs and r outputs has mr programmable edges, and the number of programmable edges in a network constructed of crossbar switches is the sum of the number of programmable edges of its constituent crossbar switches.

The connections which run between the switches in consecutive stages will not be included in the cost of the network because we are primarily interested in the order of complexity of the cost of nonblocking networks, and it will later be established that including the edges between switches can increase the cost by at most a constant factor.

The path–length of a nonblocking network will be defined to be the maximum number of switches that a signal must traverse from an input to an output. This definition suggests that the numbers of switches traversed between different pairs of inputs and outputs need not be the same, which is indeed possible. However, this fact does not affect our bounds.

The previous work on nonblocking networks contains several important results which are worth mentioning here. The earliest and most pioneering of these results was given over thirty years ago by Clos [7] who constructed a family of nonblocking networks, and showed that these networks have fewer crosspoints than a simple crossbar switch. The main structure from which all Clos networks are generated is a cascade of three stages, an input stage, a center stage, and an output stage, where the input stage consists of k crossbar switches of m inputs and r outputs each, the center stage consists of r switches of k inputs and k outputs each, and the output stage consists of k switches of r inputs and m outputs each. The network structure is such that each input-stage switch is connected to each center-stage switches by exactly one link and each output-stage switch is connected to each center-stage switch also by exactly one link. The network has a total of $n = mk$ inputs and outputs. Clos established that this network is nonblocking if $r \geq 2m - 1$, and showed that other nonblocking networks can be obtained by recursively decomposing the crossbar switches in the center stage.

In a subsequent work, Beneš and others (see chapter 3 in [3]) introduced the notion of a *rearrangeably* nonblocking network, or simply, rearrangeable network in which an idle input can always be connected to an idle output, but possibly by rearranging some of the existing connections. Obviously, every nonblocking network is also rearrangeably nonblocking. In [3] it was further proved that a 3-stage Clos network is rearrangeable if and only if $r \geq m$.

The switching cost of nonblocking networks was addressed by Clos [7] who showed that, if m is roughly equal to $\sqrt{n}$ and the center-stage switches of a 3-stage nonblocking network are recursively decomposed to form a $(2t+1)$-stage network then its cost becomes $\frac{m^2(2m-1)}{(m-1)}\{(5m-3)(2m-1)^{t-1} - 2m^t\}$. Cantor [5] established that this expression is bounded from above by $c_0 e^{\sqrt{n}}$ where c_0 is a constant, and $n = m^{t+1}$. This is the best upper bound which could be derived from Clos constructions, even though not all such constructions have been exhausted to date. Cantor was able to improve on this upper bound by using a different construction whose switching cost is less than or equal to $8n\, log_2^2\, n$. It is easy to see that $c_0 e^{\sqrt{n}} > 8n\, log_2^2\, n$ for sufficiently large n.

Another common technique for obtaining upper bounds on the cost of a nonblocking network goes back to the original work of Bassalygo and Pinsker [1] and uses a counting argument to show that in a set of networks with a certain cost and which are known to contain nonblocking as well as blocking networks, the number of blocking networks is less than the cardinality of the entire set, leading to the existence of nonblocking networks. Using a counting argument, Bassalygo and Pinsker were able to show that nonblocking networks with about $67.3n\, log_2\, n$ edges exist. This bound was later improved by Bassalygo [2] to $53.4n\, log_2\, n$. Other similar results were given in [6,11]. The main shortcoming of this approach is that it does not lead to explicit constructions of nonblocking networks.

Rearrangeable networks with about $n\, log_2\, n$ cost are known to exist, and can be constructed directly by starting out with a particular kind of 3-stage Clos network—one which contains 2-input switches in its outer stages; and recursively decomposing its center stage until it reduces to have two inputs [3]. An argument due to Shannon [13] shows that any rearrangeable network must have at least order of $n\, log_2\, n$ 2-input switches, therefore the upper bound of $n\, log_2\, n$ is tight.

The complexity studies cited above focus primarily on the cost of nonblocking networks. While the cost bounds are valuable, they provide limited information; to gain more insight on nonblocking networks one must examine the relations between their switching cost and path–length. Such relations are the focus of this paper. Similar studies appeared in the literature on time-space tradeoffs of several other problems. For example, Borodin et. al [4] examined the time-space tradeoffs of sorting on non-oblivious machines. Savage and Swamy [12] presented an analysis of time-space relations of FFT algorithms. Other studies on time-space tradeoffs can be found in [8,10].

The remainder of this paper is organized as follows: In Section 2 we prove various inequalities which relate the cost, path–length, and fanout of a nonblocking network. In Section 3, we construct rearrangeable networks which satisfy the inequalities derived in Section 2. We also show that no nonblocking network which is derived from a 3-stage Clos construction is optimal when compared to these inequalities. The paper is concluded with a discussion of results and suggestions for future research in Section 4.

2. Relations between Cost and Path–length

In this section, we will answer the following questions:

- Given an n–input nonblocking network with a specified path–length, what is the minimum cost it can have?
- Given an n–input nonblocking network with a specified cost, how small can its path–length get?

In expressing the relations between cost and path–length, the following standard notation will be used. For a function $f(n)$, we shall write $f(n) = \Omega(g(n))$ to mean that there exists a positive constant c_1 and a positive integer n_0 such that $f(n) \geq c_1 g(n)$ for all $n \geq n_0$. We shall write $f(n) = O(g(n))$ to mean that there exists a positive constant c_1 and a positive integer n_0 such that $f(n) \leq c_1 g(n)$ for all $n \geq n_0$. We shall write $f(n) = \Theta(g(n))$ to mean that there exists two positive constants c_1, c_2, and a positive integer n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$.

The answer to both questions is found by using the property that a path must exist between every input and every output in a nonblocking network. Call all outputs of a stage in such a network *reachable* from an input to the network if there exists a path from that input to those outputs. In a network consisting of a single switch all outputs are reachable from a single input. In a network consisting of two stages, each stage having switches of f inputs, at most f^2 outputs of the second stage may be reachable from a single input (if each output of the switch that the input enters in the first stage is connected to a different switch in the second stage). Extending this to S_n stages, we have

$$n \leq f^{S_n}, \tag{1}$$

where n is the number of inputs to the network. In fact, in all but trivial cases (such as a crossbar switch) the inequality is strict. To prove this, we need the following statement.

Definition 1: A multistage network in which no pair of input and output is connected by more than one path is called a *unique-path* network.

Theorem 1: A unique-path network of more than one stage is blocking.

Proof: Suppose a unique-path network can realize a permutation in which there are two disjoint (not going through any common switches) paths, one path going from input a to output α and the other going from input b to output β. Consider now a second permutation, like the first, except that a is mapped to β and b is mapped to α. At least one switch in the path from a to α will be in the path from a to β. Examine the last switch which is common to the a to α path and the a to β path. Let c denote the output of that switch which goes to β. Since the path between a and α, and b and β do not have any switches in common, the path which c was part of in the first permutation is blocked in the second permutation by the path from a to β. Therefore the second permutation is not realizable and the network is blocking. $||$

Now suppose (1) had an equal sign, that is, suppose there existed a nonblocking network with exactly $log_f\, n$ stages for $n > f$. Then no two switches in that network can be connected by more than one path, since otherwise the assumption that $n = f^{S_n}$ would not hold. Thus, a path between an input and output, if it exists at all, is unique. But Theorem 1 asserts that such a network cannot be nonblocking, leading to a contradiction. Therefore, we conclude that, for any nonblocking network, n must be strictly greater than f^{S_n} for $S_n > 1$.

By counting the programmable edges in a nonblocking network composed of S_n stages of $f \times f$ switches, we find its cost to be

$$C_n = S_n \frac{n}{f} f^2 = S_n n f \tag{2}$$

edges. Combining (1) and (2), we have

$$C_n \geq S_n n^{1+1/S_n}. \tag{3}$$

This inequality assumes that all switches in a nonblocking network are of the same size. Suppose the sizes of the switches are different at each stage, so that, at stage i, the switches will be of size f_i. The cost of such a network would at least be

$$f_1^2\left(\frac{n}{f_1}\right) + f_2^2\left(\frac{n}{f_2}\right) + \ldots + f_s^2\left(\frac{n}{f_{S_n}}\right) = n(f_1 + f_2 + \ldots + f_{S_n}) \tag{4}$$

edges, where $f_1 \times f_2 \times \ldots \times f_{S_n} \geq n$. It can be shown that (4) is minimum when all of the f_i are equal, so that despite the uniform switch assumption, (3) is still the tightest lower bound.

Inequality (3) further assumes that nonblocking networks are constructed with square crossbar switches, i.e., those which have the same number of inputs and outputs. This restricts the use of the inequality as most of the known nonblocking networks are constructed out of non-square switches. To extend the inequality to these networks, suppose that the ith stage of a nonblocking network with S_n stages consists of $f_{2i-1} \times f_{2i}$ switches, $i = 1, 2, \ldots, S_n$. By a direct count of programmable edges, it can be shown that

$$C_n = \sum_{i=1}^{S_n}\left(\frac{n}{f_1}\frac{f_2}{f_3}\frac{f_4}{f_5}\cdots\frac{f_{2i-2}}{f_{2i-1}}\right) f_{2i-1} \times f_{2i}. \tag{5}$$

To get a lower bound on C_n we further note that the ith stage must consist of at least $\frac{n}{f_{2i-1}}$ switches, and therefore $\frac{n}{f_{2i-1}} f_{2i-1} \times f_{2i}$ or $n \times f_{2i}$ edges. Summing these edges over all the stages, we have

$$C_n \geq n(f_2 + f_4 + f_6 \ldots + f_{2S_n}). \tag{6}$$

As before, it is easy to see that the right hand side of the above inequality is minimum at $f_2 = f_4 = f_6 = \ldots = f_{2S_n}$, when subject to the condition $f_2 \times f_4 \times \ldots \times f_{2S_n} \geq n$. Letting $f_2 = f_4 = f_6 = \ldots = f_{2S_n}$, leads to $f \geq n^{\frac{1}{S_n}}$ and $C_n \geq S_n n f$, or $C_n \geq S_n n^{1+\frac{1}{S_n}}$ as in (3). Thus, the same inequality applies to nonblocking networks which are constructed with non-square crossbar switches.

Finally, note that the cost expression in (2) does not include the edges which interconnect the crossbar switches in consecutive stages. These edges are characteristically distinct from the edges in the crossbar switches in that they are permanently wired and cannot be altered. As such, the cost of these edges is left out in complexity computations reported in the literature. However, even if these edges are included, the order of complexity of C_n remains the same since, adding these edges to the right hand side of (2),

$$C_n = S_n n f + \frac{n}{f} f(S_n - 1) = S_n n(f+1) - n$$

which is $O(S_n n f)$ for large f, or large S_n.

It follows that any nonblocking network of path–length S_n must have $\Omega(S_n n^{1+\frac{1}{S_n}})$ cost. For $S_n = 1$, the right hand side of inequality (3) reduces to n^2, the cost of a crossbar switch. Furthermore, it is easily seen that C_n is minimum when $S_n = \ln n$. Therefore, we have

Theorem 2: Any nonblocking network with path–length $S_n = 1$ must have $\Omega(n^2)$ cost, and any nonblocking network with $\Theta(\ln n)$ path–length must have $\Omega(n \ln n)$ cost. ||.

We note that the second part of the statement is not very strong as a more general argument due to Shannon states that any nonblocking network must have $\Omega(n \ln n)$ cost regardless of its path–length.

An expression for cost, C_n, vs. fanout, f, can be derived directly by solving (1) for S_n and substituting it into (2):

$$C_n \geq nf \log_f n. \tag{7}$$

Therefore, the following theorem holds:

Theorem 3: Any nonblocking network with $\Theta(1)$ fanout (except $f = 1$) must have $\Omega(n \ln n)$ cost, and any nonblocking network with $\Theta(n)$ fanout must have $\Omega(n^2)$ cost. ||.

These results provide lower bounds on the cost of a nonblocking network with a fixed path–length, or fixed fanout. Similar bounds can be obtained for the path–length of a nonblocking network with a fixed fanout and cost by noting that Equation (1) can be used to conclude that $S_n \geq \log_f n$. Combining Shannon's bound on the cost of a network, i.e., $C_n \geq \frac{n}{f} \log_f n$ with this latter equation, we obtain,

$$C_n S_n \geq \frac{n}{f} \log_f^2 n \tag{8}$$

or,

$$S_n \geq \frac{n \log_f^2 n}{f C_n}. \tag{9}$$

It follows that

Theorem 4: Any nonblocking network with $\Theta(n \log_f n)$ cost must have $\Omega(\frac{\log_f n}{f})$ path–length, and any nonblocking network with $\Theta(n^2)$ cost must have $\Omega(\frac{\log_f^2 n}{fn})$ path–length. ||.

3. Near-optimal Nonblocking Networks

Inequalities (3), (7), (9) underscore the relations between the cost, path–length and fanout of a nonblocking network and as such they govern the tradeoffs between these parameters. Inequalities (3) and (7) specify lower bounds on the cost of a network for a given path–length and fanout, while (9) provides on its path–length for a given cost and fanout. Consequently, any nonblocking network for which the ratio of the left and right hand sides of these inequalities is a constant can be viewed as optimal or *near optimal*. For example, both crossbar switches and Benes network are optimal rearrangeable networks, the former with constant path–length, and the latter with asymptotically minimum cost.

In addition to these two networks, we can obtain a family of nearly optimal rearrangeable networks directly from the family of Clos networks by appropriately fixing m, i.e., the number of inputs to the crossbar switches in the outer stage as given below.

The cost of an n-input rearrangeable Clos network can be expressed recursively as

$$C_n = 2nm + mC_{n/m}$$

If the center stage switches are recursively decomposed into rearrangeable Clos networks t times, then it can be shown that

$$C_n = 2nmt + m^t C_{n/m^t}$$

where $t = \frac{S_n - 1}{2}$ and C_{n/m^t} is the cost of each crossbar switch in the center stage of the decomposed network. Since these switches in the center stage are not decomposed any further, we can let $C_{n/m^t} = n^2/m^{2t}$, so that the above expression becomes

$$C_n = 2nmt + \frac{n^2}{m^t}.$$

By substituting $t = \frac{S_n - 1}{2}$, we obtain

$$C_n = nm(S_n - 1) + \frac{n^2}{m^{\frac{S_n - 1}{2}}}.$$

It can be shown that C_n is approximately minimum when $m = \Theta(n^{2/S_n})$. Upon replacing m by n^{2/S_n} on the right hand side, we have

$$C_n = (S_n - 1)n^{1+\frac{2}{S_n}} + \frac{n^2}{n^{\frac{S_n - 1}{S_n}}},$$

or,

$$C_n = (S_n - 1)n^{1+\frac{2}{S_n}} + n^{1+\frac{1}{S_n}}.$$

It can be seen that these networks are almost optimal; in particular, for large values of S_n, the expression on the right hand side approaches that in inequality (3).

As for nonblocking networks, we note that a switch is an instance of an optimal nonblocking network. However, unlike in rearrangeable networks, there is no known explicit construction which leads to optimal or near optimal nonblocking networks in the sense of (3). In fact, in what follows we establish that nonblocking Clos networks are not optimal or near optimal.

To do this, we note that the cost of an n-input nonblocking Clos network can be expressed recursively as

$$C_n = 2n(2m-1) + (2m-1)C_{n/m}.$$

If the center stage switches are recursively decomposed into nonblocking Clos networks t times, then it can be shown that

$$C_n = \frac{2nm(2m-1)}{m-1}\left[\left(\frac{2m-1}{m}\right)^t - 1\right] + (2m-1)^t C_{n/m^t}$$

where $t = \frac{S_n - 1}{2}$ and C_{n/m^t} is the cost of each crossbar switch in the center stage of the decomposed network. As before, we can let $C_{n/m^t} = n^2/m^{2t}$, so that the above expression becomes

$$C_n = \frac{2nm(2m-1)}{m-1}\left[\left(\frac{2m-1}{m}\right)^t - 1\right] + \frac{(2m-1)^t n^2}{m^{2t}}.$$

As an approximation, $(2m-1)/m$ will be replaced by 2, and $(2m-1)/(m-1)$ will be replaced by 3, yielding the upper bound

$$C_n \leq 6nm(2^t - 1) + \left(\frac{2}{m}\right)^t n^2.$$

Upon substituting $t = \frac{S_n - 1}{2}$,

$$C_n \leq 6nm\left(2^{(S_n-1)/2} - 1\right) + \left(\frac{2}{m}\right)^{(S_n-1)/2} n^2.$$

It can be shown that C_n is approximately minimum when $m = \Theta(n^{2/S_n})$. Upon replacing m by n^{2/S_n} on the right hand side, we have

$$C_n \leq 6n^{2/S_n+1}\left(2^{(S_n-1)/2} - 1\right) + \left(\frac{2}{n^{2/S_n}}\right)^{(S_n-1)/2} n^2,$$

or,

$$C_n = \Theta\left[6n^{2/S_n+1}\left(2^{(S_n-1)/2} - 1\right) + n2^{(S_n-1)/2}\right].$$

Upon comparing this with (3) we find that nonblocking Clos networks are not optimal.

4. Concluding Remarks

The paper has presented a complexity analysis for nonblocking switching networks. Inequalities relating the cost, path–length and fanout of rearrangeable and nonblocking networks have been derived. In the case of rearrangeable networks, it has been shown that Clos networks can be minimized to yield near optimal rearrangeable networks. For nonblocking networks, it has been shown that Clos networks are not optimal. Therefore, it remains to be shown whether the bound on cost given in (3) is not tight, or there exists a nonblocking Clos network which is optimal.

REFERENCES

[1] L. A. Bassalygo and M. S. Pinsker, "Complexity of Optimum nonblocking switching without reconnections," *Problems of Information Transmission*, Vol. 9, No. 1, 1974, pp. 64-66.

[2] L. A. Bassalygo, "Asymptotically optimal switching circuits," *Problems of Information Transmission*, Vol. 17, No. 3, 1981, pp.81-88.

[3] V.E. Beneš , *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press Pub. Company, New York, 1965.

[4] A. Borodin et. al, "A time-space tradeoff for sorting on nonoblivious machines," *IEEE Conference on Foundations of Computer Science*, pp. 367-378, 1979.

[5] D. Cantor, "On nonblocking networks," *Networks*, Vol. 1, pp. 367-378, 1972.

[6] F. R. K. Chung, "On concentrators, superconcentrators, generalizers, and nonblocking networks," *The Bell Systems Technical Journal*, Vol 58, No. 8, Oct. 1978, pp. 1765–1777.

[7] C. Clos, "A study of non–blocking switching networks," *The Bell System Technical Journal*, March 1953, pp. 406–424.

[8] T Lengauer, and R. Tarjan, "Upper and lower bounds on time-space tradeoffs," *in Proc. ACM Symposium on Theory of Computing*, 1979, pp. 262-277.

[9] M. J. Marcus, "The theory of connecting networks and their complexity: a review," *Proceedings of the IEEE*, Vol. 65, No. 9, September 1977, pp. 1263–1271.

[10] W. Paul, and R. Tarjan, "Time-space tradeoffs in a pebble game," *Lecture Notes in Computer Science*, Springer-Verlag, 1977.

[11] N. Pippenger, "On rearrangeable and nonblocking switching networks," *Journal of Computer and Information Sciences*, Vol. 17, 1978, pp. 145–162.

[12] J. Savage, and S Swamy, "Space-time tradeoffs on the FTT algorithms," *IEEE Transactions on Information Theory*, Sep. 1978, 563-568.

[13] C. Shannon, "Memory requirements in a telephone exchange," *The Bell System Technical Journal*, Vol. 29, 1950, pp. 343–349.

ON SELF-ROUTABILITY OF THREE-STAGE SWITCHING NETWORKS[†]

BARRY G. DOUGLASS
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

A. YAVUZ ORUÇ
Electrical Engineering Department
and
Institute of Advanced Computer Studies
University of Maryland
College Park, MD 20740

ABSTRACT

A self-routing network is a switching network where the routing of each switch can be determined as a function of the destination addresses of its inputs, without regard to the routing of other switches it may have. One family of networks which have been considered in the literature for self-routing in parallel computers are Clos networks. Earlier studies indicate that certain permutations can be self-routed on some Clos networks. This paper proves that the only category of Clos networks which can be self-routed for all permutations are those which have one or two switches in their outer stages. It then introduces a new routing scheme, called *weak local routing* which minimizes the set-up time overhead by keeping the destination information as local as possible. It is shown that an N-input network which exhibits weak local routing can be formed with $O(N \log^2 N)$ switches and $O(\log^3 N)$ propagation delay.

1 Introduction

This paper focuses on self-routing switching networks. A switching network is called *self-routing* if each of its switches can determine its own state only from the destination addresses of its own inputs, regardless of the destination addresses of the inputs of the other switches in the network. Such networks find applications in telephone switching and interprocessor communications in parallel computers [Pipp82, MaGN79, Feng81]. An alternative to self-routing networks is globally-routed networks such as the Benes network [Bene65]. However, global routing either requires too much time or too much hardware; for example, on a single processor, the Benes network needs $O(N \log N)$ steps to program for N inputs even though it has only $O(\log N)$ path–length [Waks68, OpTs71], and to reduce the time to $O(\log^2 N)$, one needs a parallel computer with N processors which are interconnected by graphs most of which are even more costly than the Benes network itself [NaSa82, LePV81, CaOr87].

This inherent difficulty with global routing has prompted research into the self-routing aspect of Clos networks. Nassimi and Sahni established that many permutations frequently used in parallel computations, which they named *class F*, can be self-routed through the Benes network [NaSa81]. Recently, Boppana and Raghevandra showed that many more permutations which they named *class L* can also be self-routed by the same network [BoRa88].

[†] This work is supported in part by the National Science Foundation under Grant No: CCR-8708864.

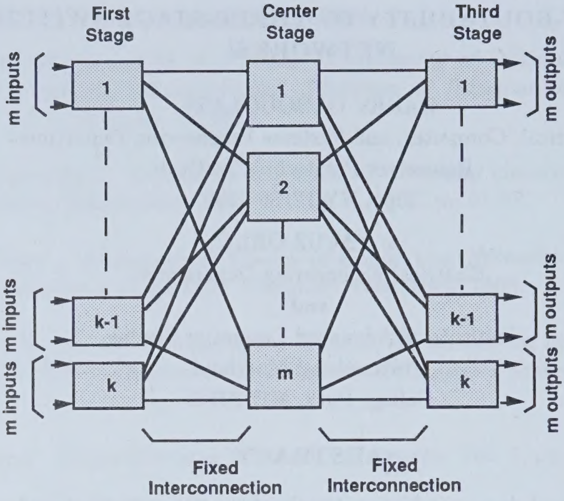


Figure 1: An N-input Clos network.

Given these results, a question naturally arises as to whether the Benes network, or more generally Clos networks can self-route all permutations. The main result of this paper is the answer to this question in the negative. It is shown that all Benes networks with more than five inputs are not self-routing. More generally, it is shown that no Clos network whose first stage contains more than two switches is self-routing. With these results established, the degree of locality of routing in these networks is examined, and a new notion of routing, called *weak local routing* is introduced. A rearrangeable network possessing weak local routing with $O(N \log^2 N)$ 2-input logic gates and $O(\log^3 N)$ delay is described.

Section 2 presents the proof that Clos networks with more than five inputs are not self-routing. Section 3 presents the main elements of weak local routing, and describes a network which exhibits weak local routing. The paper is concluded in Section 4.

2 Self-routability of Clos Networks

An N-input Clos network [Clos53], where $N = mk$, consists of three stages as depicted in Figure 1. The first and third stages of the network each consist of k $m \times m$ switches which may be implemented as crossbars, or by other permutation networks and this may be recursively repeated. The center stage consists of m $k \times k$ switches similarly implemented. The network has exactly one link between every two switches in its consecutive stages. The switches in the first stage will be denoted throughout the paper by IS_i, those in the third stage will be denoted by OS_i where $1 \leq i \leq k$, and the switches in the center stage will be denoted MS_i where $1 \leq i \leq m$. It is known that this network can realize all permutations of its inputs onto its outputs [Benes65]. The networks we consider in this paper all have this same structure, but possibly for different values of m and k. In particular, the Benes network has $m = 2$, and $k = N/2$.

First, consider the self-routability of Benes networks. One easily sees that the Benes network with a single input, i.e., a single wire, or with two inputs, i.e., a 2-input crossbar are both self-routing. It can also be seen that the 3-input Benes network is self-routing since the third input need not be paired with the other two inputs in the first stage, and directly enters the center stage from which it can be self-routed since the center and third stages of the network form a unique path structure. With a little more thought, it can be shown that the 4-input Benes network is also self-routing. As for the 5-input Benes network, if the fifth input and output are connected to both switches in the center stage, then it can be shown that this network is also self-routing.

We establish that these are the only Benes networks which can be self-routed. Let Σ_N denote the set of all permutations on N inputs. This is carried out by first observing that the number of inputs which are mapped to each half of outputs in such a network is constant over all permutations in Σ_N. We then classify the types of switches in the first stage according to how their inputs are distributed to each half of outputs under permutations in Σ_N. Next we prove that for all even $N \geq 8$, there exist permutations in Σ_N which, when modified in a certain way, forces the number of inputs mapped to the halves of outputs through a center-stage switch to change. This then leads to the proof that the Benes network is not self-routable for all even $N \geq 8$. What remains to be considered is the case for $N = 6$, and networks with odd numbers of inputs which we will handle separately.

Let R_1 denote the set of outputs of the first $\lceil N/4 \rceil$ switches and R_2 denote the set of the next $\lfloor N/4 \rfloor$ switches in the third stage of an N-input Benes network. Let $\mu^p_{i,j}$ denote the number of inputs which are mapped into R_j by permutation p through center-stage switches MS_i where $1 \leq i, j \leq 2$.

Proposition 1: $\mu^p_{i,1} = \lceil N/4 \rceil$ and $\mu^p_{i,2} = \lfloor N/4 \rfloor$for all $p \in \Sigma_N$ and $1 \leq i \leq 2$.

Proof: Each center stage switch has $N/2$ output links, one to each of the $N/2$ third stage switches.$\lceil N/2 \rceil$ of these are connected to switches $OS_1, OS_2, \ldots, OS_{\lceil N/4 \rceil}$ whose outputs define R_1, and the remaining $\lfloor N/4 \rfloor$ to switches $OS_{\lceil N/4 \rceil+1}, OS_{\lceil N/4 \rceil+2}, \ldots, OS_{N/2}$ whose outputs define R_2. Since all N output links between the center and third stage switches must be occupied to realize any permutation in Σ_N, each $p \in \Sigma_N$ maps $\lceil N/4 \rceil$ inputs into R_1 and $\lfloor N/4 \rfloor$ inputs into R_2 through the $N/2$ output links of each of the center stage switches. ||

Let $S^p_{i,j}$ denote the number of switches in the first stage of an N-input Benes network one input of which is mapped to R_i and the other input of which is mapped to R_j by $p \in \Sigma_N$ where $1 \leq i \leq j \leq 2$. Call the switches for which $i = j = 1$, switches of *type 1*, those for which $i = j = 2$, switches of *type 2*, and those for which $i = 1, j = 2$, switches of *type 3*. The following statements hold.

Proposition 2: For all $N \geq 8$ there exists a permutation $p \in \Sigma_N$ for which $S^p_{1,2} \geq 3$.

Proof: For $N \geq 8$, the first stage contains at least four switches. Choose p so that at least three of these switches $IS_h, IS_i, IS_j \in S^p_{1,2}$. Obviously, p is not unique. ||

Proposition 3: For all $N \geq 8$ there exists a permutation $p \in \Sigma_N$ which maps one input of each of at least two first-stage switches of type 3 to R_1 through one center stage switch, and maps the second input of each of these same switches to R_2 through the other center stage switch.

Proof: From Proposition 2, there exists at least four first stage-switches any subset of which can be fixed as type 3 switches by choosing an appropriate permutation in Σ_N. If at least three are fixed as type 3 switches, then, obviously, that permutation must map one input of each of at least two of these switches into R_1 through the same center-stage switch, and their other inputs into R_2 through the other center-stage switch. ||

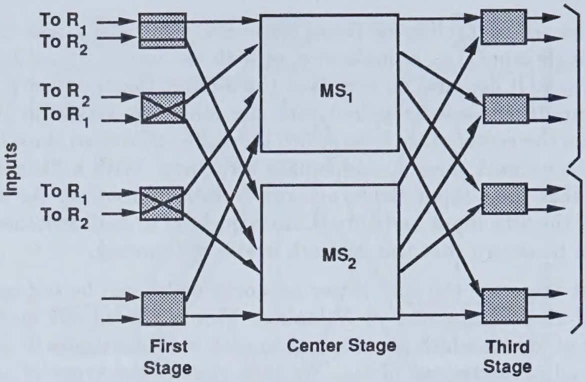


Figure 2: Type 3 switches in a Benes network.

A graphical construction of this proposition is depicted in Figure 2. It is seen that the first three switches in the first stage are fixed as type 3, and the first two map one of their inputs to R_1 through the upper center–stage switch, and their other inputs to R_2 through the lower center–stage switch. The third switch routes one of its inputs to R_1 through the lower center–stage switch and its other input to R_2 though the upper center–stage switch. The fourth switch is left unspecified, even though it should also be of type 3 as implied by the following proposition.

Proposition 4: $|R_i| = 2S^p_{i,i} + S^p_{1,2}; 1 \leq i \leq 2$, and $S^p_{1,1} - S^p_{2,2} = \lceil N/4 \rceil - \lfloor N/4 \rfloor$ for all $p \in \Sigma_N$.

Proof: That $|R_i| = 2S^p_{i,i} + S^p_{1,2}; 1 \leq i \leq 2$ is obvious since each first–stage switch of type 1 as well as tpye 2 contribute two inputs, and each first–stage switch of type 3 contributes one input to R_i. That $S^p_{1,1} - S^p_{2,2} = \lceil N/4 \rceil - \lfloor N/4 \rfloor$ follows immediately from the fact that $R_i = 2S^p_{i,i} + S^p_{1,2}; 1 \leq i \leq 2$, and $R_1 = 2\lceil N/4 \rceil$, and $R_2 = 2\lfloor N/4 \rfloor$. ||

We now state the main result of this section.

Theorem 1: No Benes network with an even number of inputs equal to, or greater than eight can be self-routed.

Proof: First by Proposition 2, construct a permutation p for which $S^p_{1,2} \geq 3$. Then by Proposition 3, we can find two first-stage switches of type 3, say IS_x and IS_y such that p maps one input from each to R_1 through the upper center-stage switch, and the other input from each to R_2 through the lower center-stage switch as shown in Figure 3. Call the inputs to the first of these two switches x_1, x_2, and the inputs to the second y_1, y_2, so that $p(x_1), p(y_1) \in R_1$, and $p(x_2), p(y_2) \in R_2$. Now construct another permutation, say $q \in \Sigma_N$ such that $q(x_2) = p(y_1)$ and $q(y_1) = p(x_2)$, and $q(x) = p(x)$ for all other inputs x over which p is defined, i.e., the remaining inputs of the network. Since only the outputs of x_2 and y_1 have changed under q, the network can accomodate this change only by redefining the states of the two switches to which these inputs are connected if it is to be self-routing. However, regardless of how the two switches are set, now both inputs to both switches are to be mapped to the same set of outputs, i.e., inputs of IS_x to R_1, and inputs of IS_y to R_2. Therefore, the number of inputs which are routed to R_1 through the upper center-stage switch decreases by one, and the number of inputs which are routed to R_1 through the lower center-stage increases by one. A similar change occurs in the

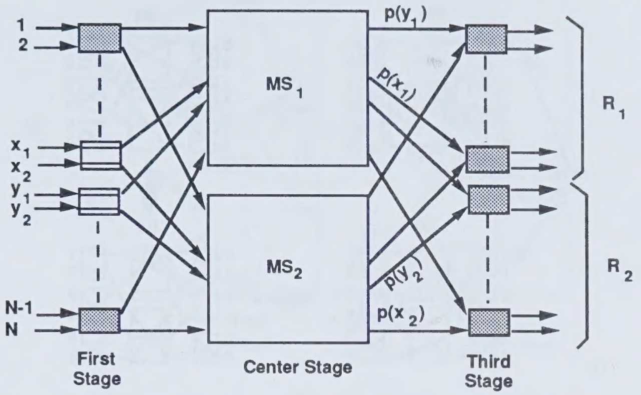


Figure 3: The partial setting of a Benes network under permutation p.

number of inputs which are routed to R_2. But this contradicts Proposition 1 which states that the number of inputs which are mapped to each half of outputs of a Benes network through each of its center-stage switches is constant for all $p \in \Sigma_N$. Hence the statement follows. ||

We now consider the remaining cases. For $N = 6$ the Benes network has three switches in each of its outer stages and two switches in the center stage. In order for the network to be self-routing, the setting of each switch must be fixed for each pattern of inputs it receives. In Figure 4 we list a sequence of permutations and their realizations on a 6-input Benes network. In Figure 4(a), switch IS_1 is arbitrarily fixed so that input 1 is routed to output 4 through switch MS_1, and input 2 is routed to output 1 through MS_2. The settings of switches IS_2 and IS_3 are then determined. In Figure 4(b), the inputs of IS_2 are kept the same as in Figure 4(a) as highlighted by shading, and therefore its setting remains the same and the settings of the other switches are then determined. Continuing in the same way, we arrive at Figure 4(d), and find that the setting of IS_1 disagrees with that in Figure 4(a) even though its pattern of inputs remains the same. We conclude that the network is not self-routing.

As for odd values of $N \geq 7$, it is easily seen that a Benes network with an odd number of inputs can be obtained directly from the network with $N-1$ inputs by isolating an input and output, and connecting them to both of the center stage switches. The proof that a network with odd number of inputs is not self-routable is then an immediate consequence of the proof that the network with one less inputs is not self-routable.

Thus, we have shown:

Theorem 2: An N-input Benes network is self-routing if and only if $N \leq 5$.

These statements can be extended to Clos networks as follows. First, we divide the outputs into two halves R_1 and R_2 as before, and extend the definition of a type 3 switch as one which sends exactly $m/2$ inputs to R_1, and $m/2$ inputs to R_2. Next, we identify a switch, say MS_i in the center stage. Then we note that Proposition 2 still holds for the extended type 3 switch. Therefore, for $k \geq 4$, we can construct a permutation so as to have at least three type 3 switches. Furthermore, along the lines of Proposition 3, we can construct a permutation which sends one input of each of at least two first-stage

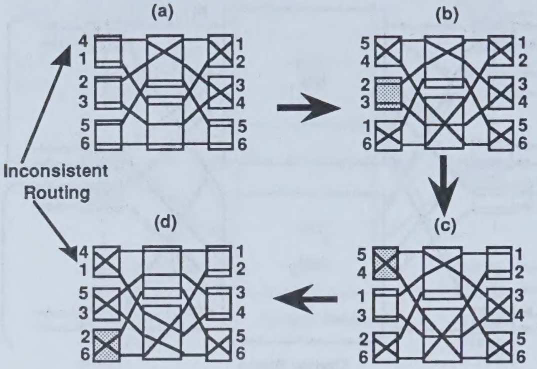


Figure 4: The sequence of routings for the $N = 6$ Benes network.

switches of type 3 to R_1 through MS_i. With these statements, we can now prove the following statement.

Theorem 3: No Clos network with $k \geq 4$, and even $m \geq 2$ is self-routing.
Proof: As specified above, choose permutation, p so that MS_i receives one input from a first-stage switch IS_x of type 3 and one input from another first-stage switch IS_y of type 3 both going to R_1. Now, by interchanging the inputs of these two switches only, construct a permutation, q, under which all inputs of IS_x go to R_1, and all inputs of IS_y go to R_2. With this change, regardless of how the inputs of the two switches are routed, MS_i must send one input from IS_x to R_1, and one input from IS_y to R_2. But, this requires that the number of inputs which are mapped to R_1 through MS_i decrease by one, and the number of inputs which are mapped to R_2 through MS_i increase by one. However, this contradicts the fact that the number of paths from MS_i to R_1 and R_2 is fixed. ||

There are two cases which remain to be considered. That Clos networks with odd values of m cannot be self-routed immediately follows from the above theorem. The case for $k = 3$ can be proven by constructing a counter example which is analogous to that in Figure 4, and will be omitted here.

Combining the above statements we now have

Theorem 4: No Clos network with $k \geq 3$, and $m \geq 2$ is self-routing.

3 Weak Local Routing

The results of the preceding section establish that no routing scheme can ever get away from coupling the destination information at individual switches in the first stage of a Clos network. Nonetheless, this coupling is structured enough to suggest a routing scheme which may keep the exchange of destination information among the switches to a minimum. The theorems of the previous section precisely define this coupling in terms of two regions of third-stage switches of a Clos network. If a new network is defined in which the switches in each of the two regions are combined into a single switch, then a routing scheme can be developed to simplify the coupling of information among the switches of the first stage.

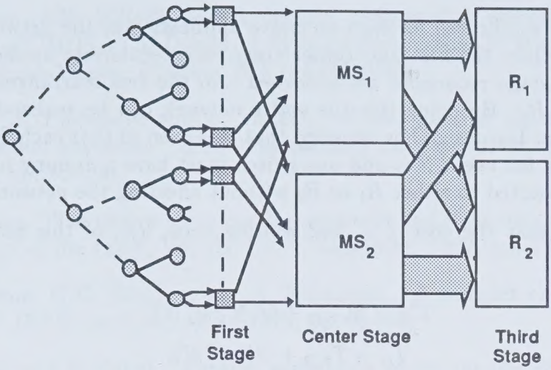


Figure 5: The routing scheme for the modified Benes network.

In particular, the Benes network can be modified by replacing the switches in the third stage of the Benes network by two switches each receiving half of the inputs from the center stage as shown in Figure 5. The routing scheme is designed so as to couple the first stage switches by a simple binary tree of comparators. The purpose of the tree is to insure that the conditions of Proposition 4 are always met. Suppose that $N = 4n$ for $n \geq 2$. Since both the inputs of type 1 and type 2 switches are mapped to the same half of the outputs, they can be set either to pass their inputs straight through, or interchange them. As for the switches whose inputs are mapped to different halves of the outputs, i.e., type 3 switches, Propositions 1 and 4 imply that the inputs of these switches which are mapped to the same half of the outputs must be routed equally between the two center stage switches.

To accomplish this requires that half the type 3 switches each route the input whose destination is in R_1 through MS_2, while half route it through MS_1. Since type 1 and type 2 switches automatically divide their connections equally between MS_1 and MS_2, all that is needed to handle this case is a device that detects if both inputs are the same and sets the switch to identity if they are. To handle type 3 switches, the binary tree is designed to route the inputs going to R_1 equally between MS_1 and MS_2 as follows. The switches in the first stage are tied to the leaves of the tree. Each leaf examines the inputs to the switch to which it is tied. If it is a type 1 or type 2 switch it sends a 0 signal up the tree. Otherwise, a 1 signal is sent. Thereafter, at any intermediate node, wherever two 1 signals meet, two signals are generated and fed back towards the leaves; the signal going down the left branch upon reaching the leaf node instructs the switch to send the input going to R_1 through MS_1. The signal going down the right branch instructs the switch to send it through MS_2 and a 0 signal is sent to the father node. On the other hand, if an intermediate node receives one 0 signal and one 1 signal from its branches, then it passes the 1 signal up the tree to its father node and returns down the same branch the signal received from the father node. If the number of connections to R_1 and R_2 are odd each will have one extra connection from MS_1 and MS_2 respectively. In this case, a single 1 signal may reach the root node. The return signal must instruct the switch accordingly.

This tree must be duplicated for each recursive application of the network structure to MS_1 and MS_2. Only the first and center stages are recursively applied to MS_1 and MS_2. The third stage recursions are absorbed into the two rearrangeable third stage switches, R_1 and R_2. By doing this the entire network can be realized by elementary switches. Note that the design has assumed that N is even so that each first stage switch has two inputs. In the event N is odd one switch must have a dummy input, which can arbitrarily be connected to either R_1 or R_2 without affecting the network.

It can be shown that the cost, C_N, and routing time, T_N, of this network obey the recursions:

$$C_N = 2C_{N/2} + O(N \log_2 N),$$
$$T_N = T_{N/2} + O(\log_2^2 N),$$

The solutions of these recursions yield:

$$C_N = O(N \log_2^2 N),$$
$$T_N = O(\log_2^3 N).$$

4 Conclusions

This paper has established that an N-input Benes network is self-routing if and only if $N \leq 5$. More generally, it has been shown that a 3-stage Clos network with m-input switches in its outer stages, and k-input switches in its center stage is not self-routing for all $m \geq 2$ and $k \geq 3$. Given these facts, the paper has then described a weak local routing algorithm which can program an N-input Benes network in $O(\log^3 N)$ gate delays using $O(N \log^2 N)$ gates.

As for further research, it will be worthwhile to determine if this algorithm can be extended to other Clos networks. It will also be interesting to implement this algorithm in VLSI and determine if there is any performance degradation due to wire-crossings in the layout.

REFERENCES

1. V.E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press Pub. Company, New York, 1965.
2. R. Boppana and C. S. Raghevandra, “On self-routing in Benes and shuffle-exchange networks,” *Proceedings of International Conference on Parallel Processing*,, Vol. 1, Aug. 1988, pp. 196–200.
3. J.D. Carpinelli and A. Y. Oruç “Parallel set-up algorithms for Clos networks,” In *Proceedings of 2nd International Conference on Supercomputing*. May 1987, pp. 321-327, Santa Clara, CA.
4. C. Clos, “A study of non–blocking switching networks,” *The Bell System Technical Journal*, March 1953, pp. 406–424.
5. T–Y Feng, “A survey of interconnection networks,” *IEEE Computer*, Dec. 1981, pp. 12–27.

6. S.C. Knauer, J.H. O'Neill, and A. Huang, "Self-routing switching network," in *Principles of CMOS VLSI Design, A Systems Perspective,* N. Weste and K. Eshraghian, Addison-Wesley, Reading, MA, pp. 424–449.

7. G.F. Lev, N. Pippenger and L. Valiant, "A fast parallel algorithm for routing in permutation networks," *IEEE Transactions on Computers,* Vol. C–30, No. 2, Feb. 1981, pp. 93–100.

8. M.J. Marcus, "The theory of connecting networks and their complexity: a review," *Proceedings of the IEEE,* Vol. 65, No. 9, September 1977, pp. 1263–1271.

9. G.M. Masson, G.C. Gingher, and S. Nakamura, "A sampler of circuit switching networks," *IEEE Computer,* June 1979, pp. 32–48.

10. D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes permutation network," *IEEE Transactions on Computers,* Vol. C-31, No. 2, Feb. 1982, pp. 148–154.

11. D. Nassimi and S. Sahni, "A self-routing Benes network, and parallel permutation algorithms," *IEEE Transactions on Computers*, Vol. C-30, No. 2, May 1981, pp. 157–161.

12. D.C. Opferman and N.T. Tsao–Wu, "On a class of rearrangeable switching networks," *The Bell System Technical Journal,* May-June 1971, pp. 1579-1600.

13. N. Pippenger, "Telephone switching networks," *in Proc. of Symposia of Applied Mathematics,* Vol. 26, May 1982, pp. 101–113.

14. A. Waksman, "A permutation network," *Journal of the ACM,* vol. 15, No. 1, January 1968, pp. 159–163.

Randomization and Scheduling in Decentralized Detection

Peter Willett *
University of Connecticut
Storrs CT 06269

Douglas Warren
University of Illinois at Chicago
Chicago IL 60680

For *centralized* tests it can be shown that the optimal receiver operating characteristic (ROC) is concave; this is a trivial consequence of the feasibility of a randomized testing rule, which can generate a straight line on the ROC between any two achievable non-randomized tests.

Let us consider a decentralized example. Suppose that the two hypotheses are:

$$\mathbf{H}_0 : X_i = N_i \qquad \mathbf{H}_1 : X_i = N_i + s_i$$

where sensor i ($i \varepsilon \{1, 2\}$) observes only one sample, $\{N_i\}$ are i.i.d. unit-variance logistic random variables, and $s_i = 2$. With sensors quantizing the local likelihood ratios to only two levels ("yes-or-no" local decisions), it is straightforward to show that both sensor mapping strategies must be identical. It can also be shown that the optimal fusion strategy must be either an AND rule (decide for H_1 only if both sensors agree on H_1) or an OR rule (decide for H_1 if either sensor decides for H_1). In Figure 1 we show both of these ROCs; note the symmetry about the line $\beta = 1 - \alpha$.

This figure is surprising, since it shows that the optimal decentralized test, defined as the maximum of these two tests, is *not concave*. It is not possible to force the ROC to be concave using randomization, since the appropriate test would be randomized between two tests which use *different sensor mapping strategies*. The implied synchronization is between all sensors and the fusion rule, hence we call this a *scheduled* test. We note that scheduled tests, while not very desirable practically, are certainly feasible, and can be optimal. We also note that scheduling is sometimes unnecessary; in the example if s_i and N_i were mutually independent and Gaussian, the OR fusion rule would be most powerful ($\forall \alpha$).

The question next arises as to the role of randomization (at the fusion center) in decentralized testing. With full freedom to choose a sensor rule which is optimal at a given false-alarm rate, it seems counter-intuitive that randomization, which is essentially a degradation of information, should ever be helpful. We present without proof [1] the following:

Theorem 1 *Given that the likelihood ratios at all sensors are mutually independent (conditioned on the state of nature) and contain no point masses of probability, no randomized test can be optimal.*

This theorem does not imply that randomization is never necessary. If the system designer does not have full freedom, a randomized test may indeed be optimal. Consider the situation of three sensors, each receiving observations which (conditioned on the state of nature) are *i.i.d.*, and each making "yes-or-no" local decisions. The statistics on the local likelihood ratios are given by the common $\mathbf{H}_0$ density:

$$f_{H_0}(x) \;=\; c\left[\frac{1}{1+\left(k(x-.5)\right)^2} - \frac{1}{1+\left(k(x-1.5)\right)^2}\right] I(0<x<2)$$

(see Figure 2) with $I(\cdot)$ the indicator function, $k = 100$, and c the appropriate normalizing factor; the alternative density is (naturally) $f_{H_1}(x) = x f_{H_0}(x)$. For the case that the local decision rules are forced to be identical, we show in Figures 3 and 4 respectively the common local threshold and the fusion rule's randomization parameter, for the optimal (constrained) test. It is perhaps surprising, but despite the sensor data being *i.i.d.*, it is possible to improve performance by using non-identical local rules. In Figure 5 we show the three *optimal* thresholds; the corresponding fusion rule does not randomize. In Figure 6 we show the ROCs of the optimal and constrained tests (the former is slightly better, although the two appear indistinguishable); it can be shown from Figure 3 that the constrained test exhibits a *convex* ROC for its region of randomization. We note in passing that if the likelihood ratios *do* contain point masses of probability (e.g. are discrete), then an optimal decentralized test may indeed use randomization.

[1] P. Willett and D. Warren, "The Suboptimality of Randomized Tests in Quantized Detection Systems", *submitted to IEEE Trans. on IT*, October 1989.

---

*Supported in part by the National Science Foundation under Grant ECS-8719031.

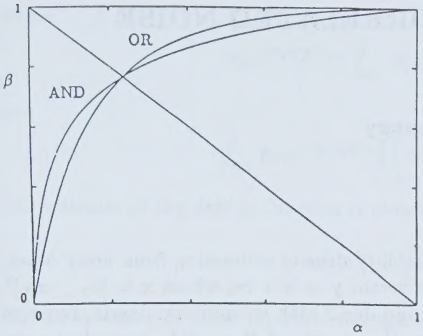


Figure (1) – ROC for Logistic Noise

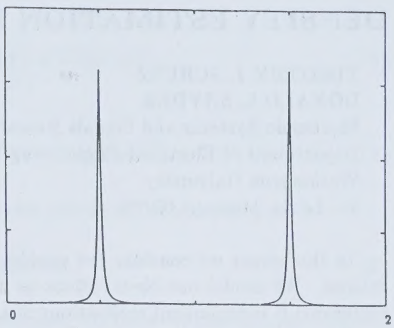


Figure (2) – Density of LR under H_0

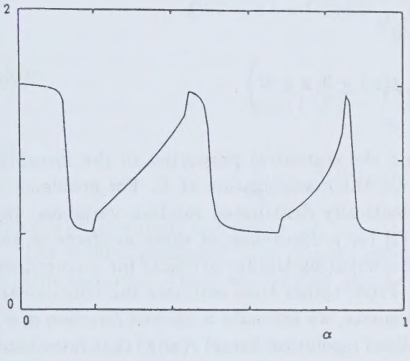


Figure (3) – Constrained Threshold

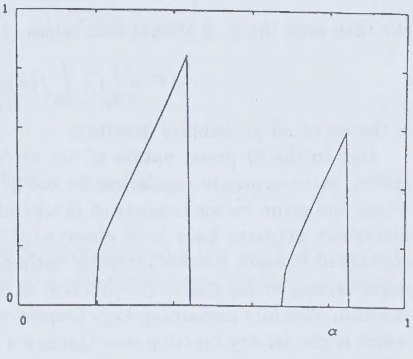


Figure (4) – Constrained Randomization

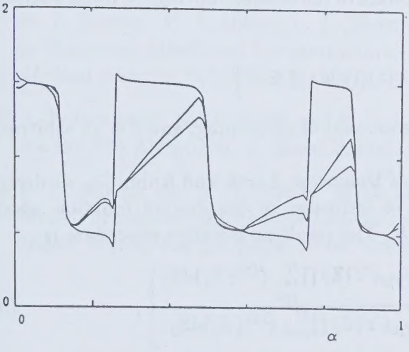


Figure (5) – Optimal Thresholds

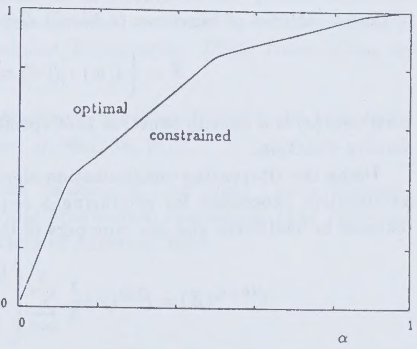


Figure (6) – ROC

DENSITY ESTIMATION IN CORRELATED NOISE [1]

TIMOTHY J. SCHULZ
DONALD L. SNYDER
Electronic Systems and Signals Research Laboratory
Department of Electrical Engineering
Washington University
St. Louis, Missouri 63130

In this paper we consider the problem of probability density estimation from noisy observations. We model our observations as a random vector $\mathbf{y} = \mathbf{x} + \mathbf{m}$, where $\mathbf{x} = [x_1 \cdots x_N]^T$ represents N independent realizations of a random variable x with an unknown density function p_x. The measurement–error vector, $\mathbf{m} = [m_1 \cdots m_N]^T$, consists of N possibly dependent noise samples with a known joint–density function, $p_{\mathbf{m}}(\mathbf{M})$. We also assume that the realization vector $\mathbf{x}$ and the error vector $\mathbf{m}$ are statistically independent.

Based on the above model, the likelihood of the data as a functional of the unknown density is

$$L(p_x; \mathbf{Y}) = \int_{\Re^N} p_{\mathbf{m}}(\mathbf{Y} - \mathbf{X}) \prod_{i=1}^{N} p_x(X_i) dX_i. \tag{1}$$

We then seek the $\hat{p}_x \in \mathcal{P}$ that maximizes (1), where

$$\mathcal{P} = \left\{ f : \int_{\Re} f(x)dx = 1, f(x) \geq 0, x \in \Re \right\}, \tag{2}$$

is the set of all probability densities.

Due to the ill–posed nature of our problem and the statistical properties of the measurements, we incorporate regularization constraints into the maximization of L. For problems in which the error vector consists of independent, identically distributed random variables, two disturbing artifacts have been observed [1]. See [2] for a discussion of these artifacts in the context of Poisson–process intensity–estimation. Anticipating similar artifacts for a correlated noise vector, we regularize the problem as follows. First, rather than estimate the true density function, possibly containing high frequency components, we estimate a *desired function* $d(w)$, which is the density function seen through a normalized resolution kernel $r(w|x)$ that introduces a known degree of smoothing according to

$$d(w) = \int_{\Re} r(w|x) p_x(x) dx. \tag{3}$$

Second, in order to avoid rough estimates that are characteristic of unconstrained maximum–likelihood density–estimation, we constrain our estimate of the desired function to be an element of the smooth set of functions (a kernel sieve)

$$\mathcal{S} = \left\{ d(w) : d(w) = \int_{\Re} s(w|z) \xi(z) dz, \xi \in \mathcal{P} \right\}, \tag{4}$$

where $s(x|z)$ is a density function that specifies the amount of smoothing, and ξ is an arbitrary density–function.

Using the alternating–maximization algorithm of Dempster, Laird, and Rubin [3], we derive an iterative procedure for producing a sequence of estimates of the desired function which increase in likelihood and are members of the set $\mathcal{P}$. The resulting iterative procedure is

$$\hat{\xi}^{(k+1)}(Z) = \hat{\xi}^{(k)}(Z) \frac{1}{N} \sum_{i=1}^{N} \left[\frac{\int_{\Re^{N-1}} p_{\mathbf{y}|\mathbf{z}}(\mathbf{Y}|\mathbf{Z}) \prod_{\substack{j=1 \\ j \neq i}}^{N} \hat{\xi}^{(k)}(Z_j) dZ_j}{\int_{\Re^N} p_{\mathbf{y}|\mathbf{z}}(\mathbf{Y}|\mathbf{Z}) \prod_{j=1}^{N} \hat{\xi}^{(k)}(Z_j) dZ_j} \right], \tag{5}$$

---

[1] This work was supported by the National Science Foundation under grant MIP–8722463.

where

$$p_{\mathbf{y}|\mathbf{z}}(\mathbf{Y}|\mathbf{Z}) = \int_{\Re^N} p_{\mathbf{y}|\mathbf{w}}(\mathbf{Y}|\mathbf{W}) \prod_{i=1}^{N} s(W_i|Z_i)dW_i, \tag{6}$$

and

$$\int_{\Re^N} p_{\mathbf{y}|\mathbf{w}}(\mathbf{Y}|\mathbf{W}) \prod_{i=1}^{N} r(W_i|X_i)dW_i = p_{\mathbf{m}}(\mathbf{Y} - \mathbf{X}). \tag{7}$$

The estimate of the desired function is obtained at stable points of (5) according to

$$\hat{d}(W) = \int_{\Re} s(W|Z)\hat{\xi}(Z)dZ. \tag{8}$$

For the special case when the noise samples are independent, the error–vector density becomes

$$p_{\mathbf{m}}(\mathbf{M}) = \prod_{i=1}^{N} p_m(M_i), \tag{9}$$

and (5) reduces to

$$\hat{\xi}^{(k+1)}(Z) = \hat{\xi}^{(k)}(Z)\frac{1}{N}\sum_{i=1}^{N}\left[\frac{p_{y|z}(Y_i|Z)}{\int_{\Re} p_{y|z}(Y_i|Z)\hat{\xi}^{(k)}(Z)dZ}\right], \tag{10}$$

where

$$p_{y|z}(Y|Z) = \int_{\Re} p_{y|w}(Y|W)s(W|Z)dW, \tag{11}$$

and

$$\int_{\Re} p_{y|w}(Y|W)r(W|X)dW = p_m(Y - X), \tag{12}$$

which was previously derived in [1].

The results presented here may be extended to density estimation for multi-dimensional random variables. These results may also be applied to stochastic inverse-problems in imaging [4].

References

[1] D. L. Snyder, M. I. Miller, and T. J. Schulz. Constrained Probability–Density Estimation from Noisy Data. In *Proceeding of the 1988 Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1988. Princeton University.

[2] D. L. Snyder, M. I. Miller, L. J. Thomas, and D. G. Politte. Noise and Edge Artifacts in Maximum-Likelihood Reconstructions for Emission Tomography. *IEEE Transactions on Medical Imaging*, 6(3):228–238, 1987.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J. Royal Statist. Soc. Ser. B*, 39:1–38, 1977.

[4] D. L. Snyder and T. J. Schulz. High Resolution Imaging at Low Light Levels Through Weak Turbulence. In *Quantum Limited Imaging and Information Processing, 1989 Technical Digest Series, Vol. 13*, pages 55–58. Optical Society of America, 1989.

The CR Bounds for Joint Estimates of Independent and Proportional FM Signals

Tod E. Luginbuhl* and Peter K. Willett†
University of Connecticut
Storrs CT 06269

Abstract

The Cramer-Rao Lower Bounds on the variance of the parameter estimates of several discrete time complex sinusoids with continuous time varying frequencies are derived and calculated. Two particular signal classes are considered. The first signal class assumes that there are no relationships between the different FM signals (independent), whereas the second signal classes assumes that the ratio between the instantaneous frequencies of any two sinusoids is an unknown constant (proportional). In both cases, the time varying frequency trajectories are approximated by continuous piecewise linear functions. The Cramer-Rao Lower Bounds are then compared when the received signal for both cases is a member of the second class of signals. Numerical calculation of the Cramer-Rao Lower Bounds show that the optimal estimator can be an order of magnitude or more better than the independent assumption estimator.

1 Introduction

There are a wide variety of problems in signal processing where it is necessary to estimate the parameters of a set of complex sinusoids. A general model for the discrete time form of these signals is

$$S_m = \sum_{l=1}^{L} A_l \exp\left[i2\pi \int_0^{mT} f_l(\tau)d\tau + i\theta_l\right]$$

where $\{A_l\}_{l=1}^{L}$ represents the set of amplitudes, $\{f_l(\tau)\}_{l=1}^{L}$ the set of continuous time varying frequencies, $\{\theta_l\}_{l=1}^{L}$ the set of constant initial phases and T the sampling period. These signals appear in communication, noise suppression, surveillance and other problems. In many of these applications, the time varying frequencies are all known to be integral multiples of some fundamental frequency which may or may not be in the set. These signals are called harmonic sets in references [5], [6] and [14]) and can be modelled using the equation above by substituting $lf(\tau)$ for $f_l(\tau)$.

In references [1] and [3] it is pointed out that the relationship between the frequencies and the fundamental in a harmonic set is not always integral. For example, a particular application

---

*Also with the Naval Underwater Systems Center, New London Laboratory, New London, CT 06320

†Supported in part by the National Science Foundation under Grant ECS-8719031

might require working with the spectrum from a bandpass filter which rejects the fundamental frequency of a set of harmonic signals but passes a number of the higher harmonics. An observer at the output of the filter might consider the lowest frequency in the harmonic set passed by the filter as the fundamental frequency of the set of signals present in the filter's pass band, and this would force the observer to use rational numbers to describe the relationship between the new fundamental and the other frequencies passed by the filter. This example is fairly typical of noise suppression problems and surveillance problems in which the frequency band of interest, or the frequency band of the sensor, is limited. Another example, which is perhaps less common but better illustrates non-integral harmonic sets, is the nonlinear vibration that results from an automobile's transmission. Since the entire mechanical system is driven by a single shaft, the rotation of every gear and shaft in the transmission is related by the gear ratios, and the gear ratios are usually rational numbers and not integers. In order to characterize the signals in the two examples as harmonic sets, a more general and rigorous definition of a harmonic set than the intuitive definition presented earlier is required.

References [1] and [3] use the following definition for harmonic sets:

Definition 1 *A harmonic set is a collection of functions, $\mathcal{H} = \{h_l(t)\}_{l=1}^{L}$ where $h_l : \mathcal{R} \to \mathcal{C}$ such that*

$$\frac{\phi_l^{(1)}(t)}{\phi_k^{(1)}(t)} = \alpha_{lk}$$

where $\phi^{(n)}(t) = d^n\phi/dt^n$, $\phi_i(t) = \arg(h_i(t))$, $h_i(t) \in \mathcal{H}$ and $\alpha_{lk} \in \mathcal{R}$ for all $t \in \mathcal{R}$.

By applying this definition, the spectrum at the output of the bandpass filter (or the spectrum of the transmission's vibration) in the example can be characterized as a harmonic set. The signal model for these cases is

$$S_m = \sum_{l=1}^{L} A_l \exp\left[i2\pi\alpha_l \int_0^{mT} f(\tau)d\tau + i\theta_l\right]$$

where $\alpha_1 = 1$ because one frequency must be chosen as the fundamental. In general, the choice of the fundamental is arbitrary because Definition 1 only requires that the ratio of any two time varying frequencies in the set be constant; however, the set of proportionality constants $\{\alpha_l\}_{l=1}^{L}$ will be different for different choices of the fundamental. So, for a given harmonic set with L members, there are L equivalent ways of representing the set. Choosing the set member with the lowest frequency ($\alpha_1 = 1$) is convenient notation and corresponds to physical intuition. For example, in the case of the automobile transmission, $f(\tau)$ represents the engine crankshaft if the bandwidth of the sensor is large enough.

Since the set of constants, $\{\alpha_l\}_{l=1}^{L}$, are unknown for $l > 1$, they must be estimated along with the other parameters of the harmonic set. Note that the choice of the fundamental may influence the quality of the parameter estimates. The problem of estimating these additional constants as well as the other harmonic set parameters is highly nonlinear and is significantly more difficult than estimating the harmonic set members separately. Because of this, and because fast algorithms exist for estimating groups of independent sinusoids (for example [2]), it is important to evaluate the gain in performance of an optimal estimator over these existing algorithms. If the improvement in estimation variance is insignificant or small, the optimal estimator may not be worth implementing. At this point, it is useful to define terms for the two distinct estimation schemes. The terms optimal estimator or dependence assumption will be used when referring to the optimal estimator of the harmonic signals described above; the terms suboptimal estimator or independence assumption will be used when referring to an estimation scheme which assumes that the sinusoids are unrelated. The optimal estimator and the suboptimal estimator can be compared experimentally by running side by side monte carlo trials on the same data and comparing estimate variances. A theoretical comparison can be carried out in a straight-forward manner by comparing the Cramer-Rao Lower Bounds of the optimal estimator and a general form of the existing estimators. In this case, the Cramer-Rao Lower Bounds are easier to calculate than running the experiments. In addition, the Cramer-Rao Lower Bounds provide useful insight into the estimation problem which can aid development of estimators.

2 Derivation

In many cases of interest, harmonic sets are produced by a phase locked mechanical process such as a helicopter's rotor, a household fan or an internal combustion engine. Since these sources frequently appear in practical problems, the remainder of the discussion will focus on phase locked harmonic sets. These harmonic sets can be modelled by

$$S_m = \sum_{l=1}^{L} A_l \exp\left[i2\pi\alpha_l \int_0^{mT} f(\tau)d\tau + i\theta\right] \text{ for } m = 0, \ldots, M-1,$$

where $\alpha_1 = 1$ and θ is the initial phase for the entire set. The parameters in the set

$$\{\{A_l\}_{l=1}^{L}, \theta, \{\alpha_l\}_{l=2}^{L}, f(\tau)\}$$

are generally unknown and must be estimated. In order to estimate the time varying frequency function, it is necessary to parametrize $f(\tau)$. Using the method in [7], [15], [16] and [17], it is assumed that the signal sequence $\{S_m\}_{m=0}^{M-1}$ can be decomposed into J consecutive subsequences of length K over which the time varying frequency, $f(\tau)$, is approximately linear. If $f_j = f(jKT)$ is used to denote the instantaneous frequency at the boundary of subinterval j, then $f(\tau)$ may be expressed as

$$f(jKT + \tau) = f_j + \frac{(f_{j+1} - f_j)\tau}{KT} \text{ for } 0 \le \tau \le KT \text{ and } j = 0, 1, \ldots, J-1.$$

Under this assumption $f(\tau)$ can be estimated from estimates of the sequence $\{f_j\}_{j=0}^{J}$. Hence, the parameter set for the estimation problem under the independence assumption is

$$\left\{\{A_l\}, \{\theta_l\}, \{f_{lj}\}_{j=0}^{J}\right\}_{l=1}^{L},$$

whereas, the parameter set for the estimation problem under the dependence assumption is

$$\left\{\{A_l\}_{l=1}^{L}, \theta, \{\alpha_l\}_{l=2}^{L}, \{f_j\}_{j=0}^{J}\right\}.$$

The Cramer-Rao Lower Bounds are defined as the elements of the matrix, $\mathcal{I}^{-1}$, where $\mathcal{I}$ is Fisher's information matrix [12]. The joint information, $\mathcal{I}_{ij}$, between two parameters β_i and β_j of a signal $\vec{S}$ is

$$\mathcal{I}_{ij} = E\left\{ \frac{\partial \ln\left(p(\vec{Z}|\vec{S})\right)}{\partial \beta_i} \frac{\partial \ln\left(p(\vec{Z}|\vec{S})\right)}{\partial \beta_j} \right\} = -E\left\{ \frac{\partial^2 \ln\left(p(\vec{Z}|\vec{S})\right)}{\partial \beta_i \partial \beta_j} \right\}$$

where $\vec{Z}$ is the noise corrupted version of $\vec{S}$. The signal $S_m = S_{jK+k}$ (making use of the decomposition described above where $k = m \bmod K$) is assumed to be imbedded in additive, white, zero mean complex Gaussian noise with variance σ^2. For the independent case, the joint information between parameter,

$$\beta_{\eta_j} \in \{A_\eta, \theta_\eta, f_{\eta j}\} \text{ where } \eta = 1, \ldots, L \text{ and } j = 0, \ldots, J,$$

and parameter

$$\beta_{\nu_j} \in \{A_\nu, \theta_\nu, f_{\nu j}\} \text{ where } \nu = 1, \ldots, L \text{ and } j = 0, \ldots, J$$

can be shown to be

$$\mathcal{I}_{\eta_i \nu_j} = 2\sigma^{-2} \sum_{p=\max(i-1,j-1)}^{J-1} \sum_{k=0}^{K-1} \Re\left\{ \frac{\partial S^*_{pK+k}}{\partial \beta_{\eta_i}} \frac{\partial S_{pK+k}}{\partial \beta_{\nu_j}} \right\}$$

(reference [3]). For the optimal case, the joint information, $\mathcal{I}_{\eta_i \nu_j}$, between parameters β_{η_i} and β_{ν_j} has an identical expression, but the parameters are members of the set

$$\{\{A_l\}_{l=1}^{L}, \theta, \{\alpha_l\}_{l=2}^{L}, \{f_j\}_{j=0}^{J}\}$$

(reference [3]). Unfortunately, the Cramer-Rao Lower Bounds for these two cases can not be expressed analytically without inverting a large matrix. However, the bounds may be easily calculated numerically.

3 Numerical Results

We have evaluated these Cramer-Rao Lower Bounds, and present some numerical results in the remaining sections. In the examples presented, all the harmonic sets consist of two frequency trajectories (two members), and the signals were generated in the following manner. The time varying frequencies used to generate the harmonic sets were produced by adding the carrier frequencies to a piecewise linear modulation frequency. The boundaries of the modulation function's piecewise linear intervals were taken from the output of a second order autoregressive filter in steady state driven by white noise. The filter had an average period of 0.1 hertz and unity output variance. The linear interval was 256 samples; the sampling period was 1/256 seconds; and the total number of samples was 2560. In addition, the noise variance of the complex noise process was 1.0, and the amplitudes of the two frequency trajectories were also 1.0.

Figure 1, shows a frequency versus time (interval boundary point) plot for two frequency trajectories with carrier frequencies at 10 and 20 hertz. The frequency trajectory centered about 10 hertz has a variance of 1.0. If the 10 hertz frequency trajectory is considered the fundamental by the optimal estimator, then the frequency ratio (proportionality constant) is 2.0; otherwise, if the 20 hertz frequency trajectory is considered the fundamental by the optimal estimator, then the frequency ratio is 0.5. In all of the examples, one frequency in the harmonic sets is always the 10 hertz frequency shown in figure 1; however, this 10 hertz frequency trajectory is not always considered the fundamental by the optimal estimator.

Figures 2 and 3 are plots of the Cramer-Rao Lower Bounds for amplitude A_l and initial phase θ versus the frequency ratio respectively. For the frequency ratios which are less than 1.0, the 10 hertz frequency trajectory shown in figure 1 is not considered the fundamental frequency by the optimal estimator, but for the frequency ratios greater than 1.0, the 10 hertz frequency trajectory is considered the fundamental frequency by the optimal estimator. Hence, in addition to showing the improvement achieved in theoretical estimation accuracy by the optimal estimator, figures 2 and 3 also show the optimal estimator's sensitivity to the choice of the fundamental frequency. For example take the harmonic set shown in figure 1, these frequency trajectories were used to calculate the Cramer-Rao Lower Bounds for frequency ratios equal to 0.5 and 2.0 which correspond to fundamental frequencies center at 20 or 10 hertz respectively. Figures 2 and 3 show that the amplitude and initial phase estimates are insensitive to the choice of the fundamental. However, these plots *do* show that the optimal estimator is an improvement over the suboptimal scheme. The improvement in the amplitude estimates occurs only when the frequencies are very close together (frequency ratios near 1.0). Whereas, the improvement in the estimate of the initial phase is about 1.7 orders of magnitude for widely separated frequencies and about 2.0 orders of magnitude for closely spaced frequencies.

Figure 4 shows a plot of the Cramer-Rao Lower Bound on the estimate of the frequency ratio by the optimal estimator versus the frequency ratio. This plot shows that the frequency ratio estimate is very sensitive to the choice of fundamental. In particular, the plot shows that choosing the lowest frequency observed in the harmonic set as the fundamental improves the estimate of the frequency ratio, but this choice does not improve the estimates of the frequency trajectories. Aside from the asymmetry in the plot, the magnitude of the Cramer-Rao Lower Bound on estimating the frequency ratio is unexpected. Even for very close frequencies, the Cramer-Rao Lower Bounds are less than 10^{-7}. The plot shows that the Cramer-Rao Lower Bound on the frequency ratio is a strong function of the frequency ratio itself when the frequency ratio is greater than 1.

Figures 5 through 8 show plots of the Cramer-Rao Lower Bounds on the frequency estimates for the frequency ratios equal to 0.99 (figures 5), 1.01 (figure 6), 0.5 (figure 7) and 2.0 (figure 8). The 10 hertz frequency trajectory is considered to be the fundamental frequency by the optimal estimator in figures 6 and 8. Figures 5 and 7 are the corresponding results when the 10 hertz frequency trajectory is not considered to be the fundamental frequency by the optimal estimator. Note that figures 7 and 8 correspond to the frequency trajectories in figure 1.

From figures 5 and 6, it is clear that the optimal estimator has better performance because the Cramer-Rao Lower Bounds on the frequency estimates for the optimal estimator are tighter and more stable. It also appears that the choice of the fundamental frequency makes no difference in the performance of the optimal estimator. Currently, there is no explanation as to why Cramer-

Rao Lower Bounds on the suboptimal frequency estimates for frequencies 3 through 9 vary so much during the estimation interval. It is not surprising that the suboptimal approach should have trouble for closely spaced frequencies, but the sinusoidal behavior of the Cramer-Rao Lower Bounds is surprising.

In figures 7 and 8, the Cramer-Rao Lower Bounds on the suboptimal frequency estimates have the expected convex shape which indicates that the first and last frequencies are the hardest to estimate. The Cramer-Rao Lower Bounds on the first two frequencies estimated by the optimal estimator are surprising because they are the tightest. Why this is true is not clear, but it is probably related to the bounds on the initial phase and the frequency ratio. Once again the asymmetry due to the choice of the fundamental frequency is readily apparent; however, this asymmetry does not reflect a difference in performance because figure 8 can be obtained from figure 7 by dividing through by the square of the frequency ratio (the reverse is also true). In either case, the Cramer-Rao Lower Bounds on optimal frequency estimates are better than the corresponding bounds on suboptimal frequency estimates; in figure 8, the improvement is approximately one order of magnitude.

4 Summary

To summarize the results presented in the figures, it is clearly evident that the optimal approach is theoretically superior to the suboptimal approach on all parameter estimates. The suboptimal and optimal approaches theoretically achieve comparable amplitude estimate unless the frequencies in the harmonic set are closely spaced in frequency; when the frequencies in the harmonic set are closely spaced the optimal estimator is significantly better at estimating amplitudes. The optimal estimator is superior to the suboptimal estimator for all other parameter estimates under all conditions. In addition, the optimal estimator theoretically performs best when the lowest frequency in the harmonic set is chosen as the fundamental. The optimal estimator has Cramer-Rao Lower Bounds on initial phase and frequency which are atleast an order of magnitude smaller than the suboptimal estimator's corresponding bounds. These results suggest that development of the optimal estimator for these harmonic sets is worth pursuing.

References

[1] Albert A. Gerlach, Edward L. Kunz, Wendell L. Anderson, and Kenneth D. Flowers. *On the Simulation of Harmonically Related Signals.* NRL Report 9125, Naval Research Laboratory, Washington, DC 20375-5000, June 1988.

[2] Tom Kwan and Kenneth Martin. Adaptive detection and enhancement of multiple sinusoids using a cascade iir filter. *IEEE Transactions On Circuits And Systems*, 36(7):937–947, July 1989.

[3] Tod E. Luginbuhl. *Joint Estimation of Independent and Proportional FM Signals.* Master's thesis, University of Connecticut, 1989. in preparation.

[4] D. R. A. McMahon and R. F. Barrett. Generalisation of the estimation of the frequency of a single tone in noise from the phases of discrete fourier transforms. In *Signal Processing*, pages 371–383, Elsevier Science Publishers B. V., North-Holland, June 1987.

[5] D. R. A. McMahon and R. S. Barrett. Ml estimation of the fundamental of a harmonic series. In *Proceedings of the IASTED International Symposium on Signal Processing and its Applications*, pages 333–336, IASTED, August 1987. University of Queensland.

[6] Arye Nehorai and Boaz Porat. Adaptive comb filtering for harmonic signal enhancement. *IEEE Transactions On Acoustics, Speech, and Signal Processing*, ASSP-34(5):1124–1138, October 1986.

[7] Norman L. Owsley, Joseph J. Wolcin, and John F. Law. *Multichannel Maximum A Posteriori Signal Parameter Estimation.* Technical Memorandum 801004, Naval Underwater Systems Center New London Laboratory, New London, CT 06320, January 1980.

[8] David C. Rife and Robert R. Boorstyn. Multiple tone parameter estimation from discrete-time observations. *The Bell Systems Technical Journal*, 55(6):1389–1410, November 1976.

[9] David C. Rife and Robert R. Boorstyn. Single-tone parameter estimation from discrete-time observations. *IEEE Transcations On Information Theory*, IT-20(5):591–598, September 1974.

[10] Louis L. Scharf and Howard Elliott. Aspects of dynamic programming in signal and image processing. *IEEE Transactions On Automatic Control*, AC-26(5):1018–1029, October 1981.

[11] Robert D. Short III and John P. Toomey. Detection and estimation of frequency-random signals. *IEEE Transactions On Information Theory*, IT-28(6):940–946, November 1982.

[12] Harry L. Van Trees. *Detection, Estimation, and Modulation Theory Part I.* John Wiley & Sons, 1968.

[13] Donald W. Tufts and James T. Francis. Estimation and tracking of parameters of narrow-band signals by iterative processing. *IEEE Transactions On Information Theory*, IT-23(6):742–751, November 1977.

[14] Jin-Der Wang and H. Joel Trussell. Adaptive harmonic noise cancellation with an application to distribution power line communications. *IEEE Transactions On Communications*, 36(7):875–883, July 1988.

[15] Joseph J. Wolcin. *Maximum A Posteriori Estimation of Narrowband Signal Parameters.* Technical Memorandum 791115, Naval Underwater Systems Center New London Laboratory, New London, CT 06320, June 1979.

[16] Joseph J. Wolcin. Maximum a posteriori estimation of narrow-band signal parameters. *JASA*, 68(1):174–178, July 1980.

[17] Joseph J. Wolcin. *Maximum A Posteriori Line Extraction: A Computer Program Description.* Technical Memorandum 801042, Naval Underwater Systems Center New London Laboratory, New London, CT 06320, March 1980.

Figure 1: Frequency vs Time (Frequency Ratio = 2 or 0.5)

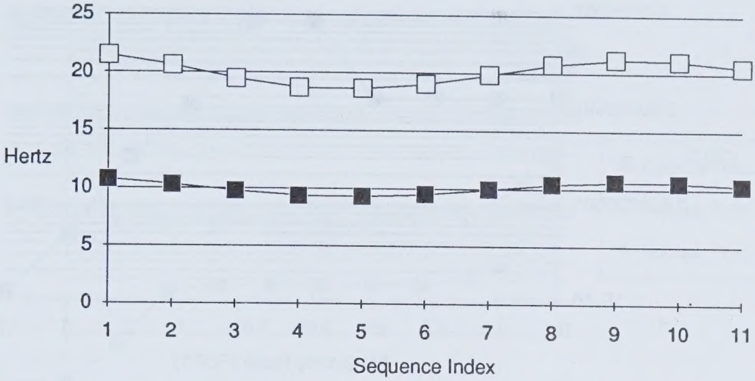


Figure 2: Amplitude Bounds vs Frequency Ratio

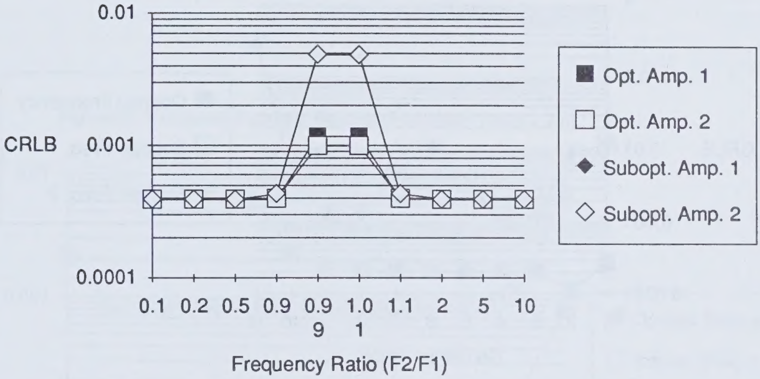


Figure 3: Initial Phase Bounds vs Frequency Ratio

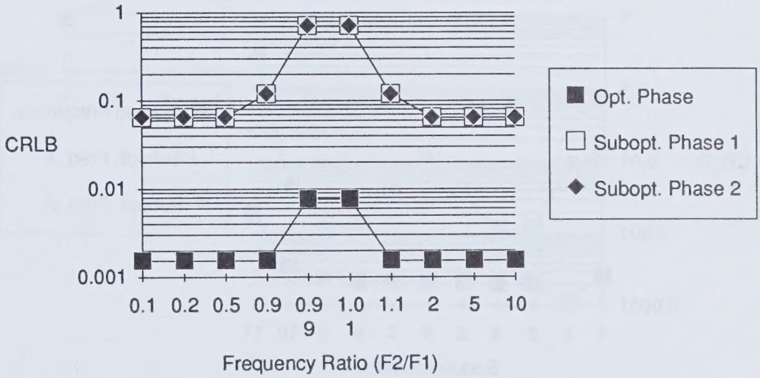


Figure 4: Frequency Ratio Bounds vs Frequency Ratio

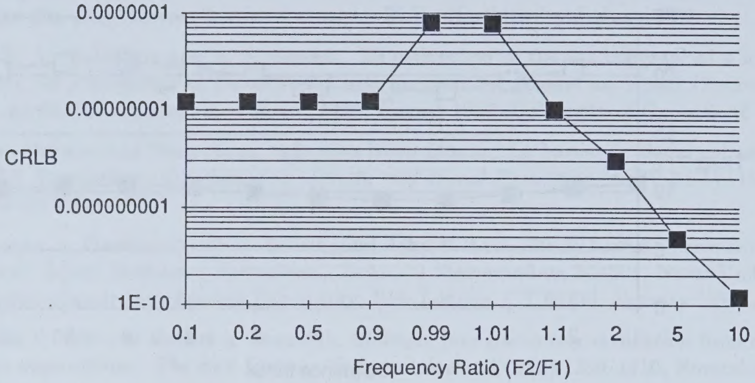


Figure 5: Frequency Estimate Bounds (Frequency Ratio = 0.99)

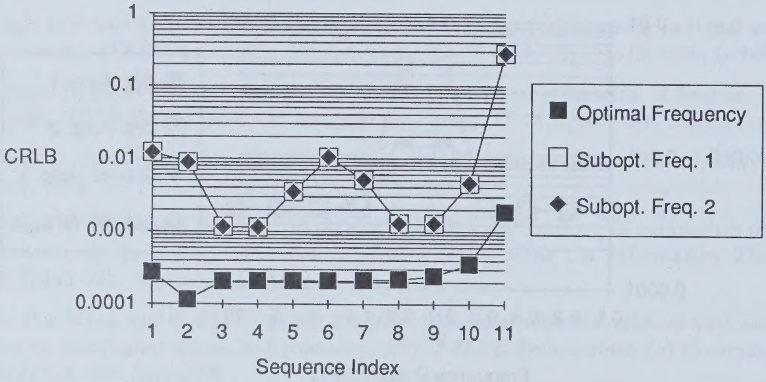


Figure 6: Frequency Estimate Bounds (Frequency Ratio = 1.01)

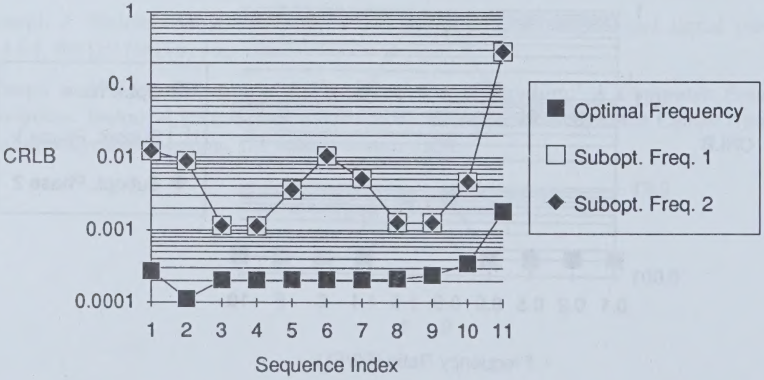


Figure 7: Frequency Estimate Bounds (Frequency Ratio = 0.5)

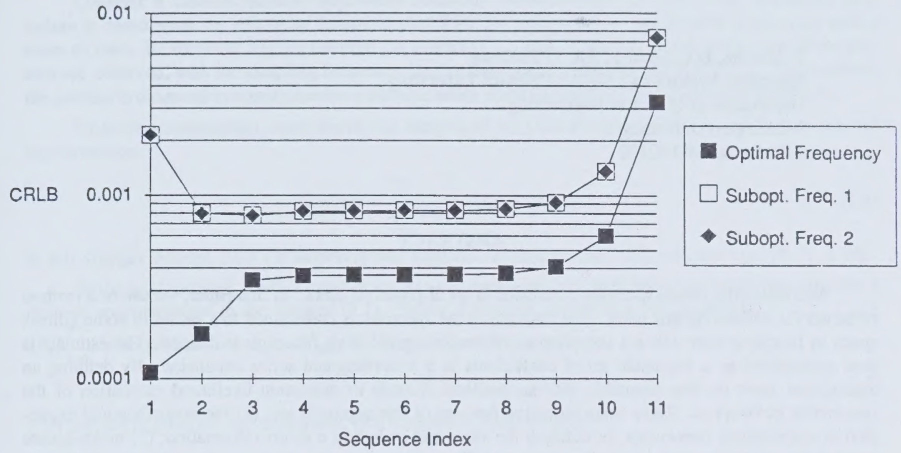
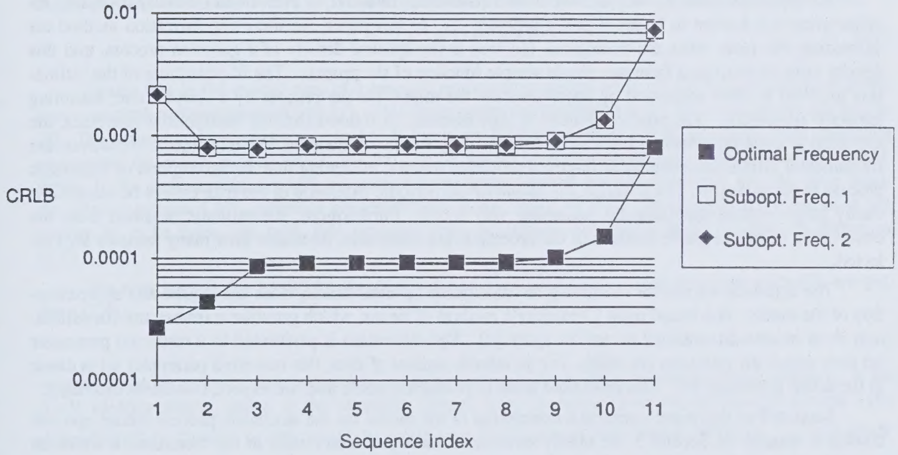


Figure 8: Frequency Estimate Bounds (Frequency Ratio = 2.0)



A SIEVE-CONSTRAINED MAXIMUM-LIKELIHOOD ESTIMATOR FOR THE SPECTRUM OF A GAUSSIAN PROCESS [*]

P. Moulin, D.L. Snyder, J.A. O'Sullivan
Electronic Systems and Signals Research Laboratory
Department of Electrical Engineering
Washington University
Saint Louis, MO 63130

ABSTRACT

Maximum-likelihood spectrum estimation is an ill-posed problem. In this paper, we use of a method of sieves for addressing this issue. The estimate of the spectrum is constrained to a subset of some Hilbert space of functions over which a complete set of nonorthogonal basis functions is defined. The estimate is then represented by a countable set of coefficients in a nonorthogonal series expansion. By defining an appropriate sieve on this countable set, our problem reduces to maximum-likelihood estimation of the parameters in the sieve. Three main attractive features of this approach are: (1) the nonorthogonal expansion is a convenient framework for defining the sieve and including *a priori* information; (2) mean-square consistency of the estimates can be expected; and (3) we have derived a tractable alternating maximization algorithm for estimating the parameters. The setup of this problem is general and can be applied without major difficulties to the estimation of higher-dimensional spectral functions, as occurs, for example, in imaging radar targets from delay-doppler data.

1. INTRODUCTION

Maximum-likelihood (ML) estimation of a continuous function, or even of an infinitely countable set of parameters is known to be an ill-posed problem [1]. In this paper, we use a regularization method for addressing this issue when the continuous function is the spectral density of a gaussian process, and this density must be estimated from one single sample function of the process. The ill-posedness of the estimation problem is often addressed by approximating the model for the process by a simpler one, assuming temporal periodicity. The spectral density is then discrete. It follows that for bandlimited processes, the spectrum is finite-dimensional and can be estimated by means of standard ML techniques. However, the fundamental difficulties inherent to ill-posed problems remain, indicating that the assumption of a periodic process is not sufficient. For instance, the length of the periodic extension of the data cannot be made arbitrarily large without the estimates becoming very rough. Furthermore, the estimate obtained from the observation of one single realization of the process is not consistent, no matter how many samples are collected.

The approach we use for estimating the continuous spectral density does not require any approximation of the model. It is based upon Grenander's method of sieves, which provides a framework for estimation in an infinite-dimensional parameter space [1]. The estimation is performed in a restricted parameter set over which the estimates are stable. For an infinite amount of data, this restricted parameter set is dense in the actual parameter set. This procedure leads to producing stable and, we expect, consistent estimates.

Section 2 of this paper contains a description of the model for the stochastic process whose spectral density is sought. In Section 3, we briefly summarize some previous results in the literature, in which an approximation of the model is made; the limitations of these methods are mentioned. Section 4 contains a short review of regularization methods. The sieve-constrained ML estimator is described in Section 5.

---

[*] The work described in this paper was supported by contract number N00014-86-K-0370 from the Office of Naval Research and by grant number MIP-8722463 from the National Science Foundation.

2. MATHEMATICAL MODEL

Consider a complex-valued, wide-sense stationary stochastic-process $y(t)$. Our estimation procedure is based upon the following statistical model for the observations. The process is gaussian with a mean of zero. Its spectrum $S(f)$ is bandlimited to some $f_{\max}$, and N samples of one realization of the process are observed, with the sampling frequency $1/\Delta t$ at least equal to the Nyquist rate $2f_{\max}$. Furthermore, the process is observed in complex-valued additive white gaussian noise $w(t)$ with spectral density N_0.

Under the assumptions stated above, the samples of $y(t)$ can be represented via the Cramer spectral representation:

$$y(n\Delta t) = \int_{-f_{\max}}^{f_{\max}} \exp(-j2\pi f n\Delta t)\, Z(df)\,. \tag{2.1}$$

In this Stieltjes integral, $Z(df)$ is an orthogonal, gaussian spectral process with variance $S(f)\,df$ [1, p.70].

In the next sections, a ML estimator for $S(f)$ is presented. We shall find it convenient to consider a discrete approximation to the integral (2.1). Define a uniform partitioning of the interval $[-f_{\max}, f_{\max}]$, $\{f_k\}_{k=0}^{M-1}$, with partition size $\Delta_M = 2f_{\max}/M$. Next define the process

$$c(k) = (\Delta_M)^{-1/2}\,[Z(f_{k+1}) - Z(f_k)]\,. \tag{2.2}$$

This process is gaussian and orthogonal. The variance of each sample,

$$\sigma^2(k) = \frac{1}{\Delta_M} \int_{f_k}^{f_{k+1}} S(f)\,df\ , \tag{2.3}$$

is an approximation to $S(f_k)$. Next, approximating the Stieltjes integral (2.1) with a Riemann sum, we get

$$\tilde{y}(n\Delta t) = \sqrt{\frac{2f_{\max}}{M}} \sum_{k=0}^{M-1} \exp(-j2\pi f_k n\Delta t)\, c(k)\,. \tag{2.4}$$

As the partition size tends to zero, $\tilde{y}(.)$ converges to the stochastic integral $y(.)$ in the mean-square sense:

$$y(n\Delta t) = \underset{M\to\infty}{l.i.m.}\ \tilde{y}(n\Delta t) \tag{2.5}$$

Notice that in the special case of a periodic process with period $M\Delta t$, $\tilde{y}(.) = y(.)$, so that (2.4) is the *exact* spectral representation of $y(n\Delta t)$. This observation has motivated using (2.4) as a model for $y(.)$ in [2]. The parameter M is greater or equal to N but is not further specified, but M should be large enough to ensure that the model approximation is valid. In [2] the issue of ill-posedness is addressed by keeping M finite. In the next section, we will summarize some known results for this estimator. On the other hand, the sieve-constrained ML estimator presented here is still based on the representation (2.4), but does not require M to be finite. As such, this estimator does not involve any model approximation, as shown in (2.5).

We now specify the model for the observations. Using the same vector notations as in [2], we get, from (2.4):

$$r = \Gamma^{\dagger} c + w\ , \tag{2.6}$$

where r is the N-vector of observations, w is a noise vector with diagonal covariance $N_0 I_N$, where I_N is the $N\times N$ identity matrix, and c is the M-vector formed from the samples of the orthogonal process $c(k)$ defined in (2.1). Since c is orthogonal, its covariance Σ is diagonal with entries $\{\sigma^2(k)\}_{k=0}^{M-1}$. Γ is a $M\times N$ matrix with (k,n) entry given by $\sqrt{\frac{2f_{\max}}{M}} \exp(-j2\pi f_k n\Delta t)$; finally, $\dagger$ denotes the Hermitian-transpose operator for matrices *.

---

* Γ can be generalized readily to accomodate a linear transformation of $y(.)$ in the measurement, as occurs in the radar imaging problem [2].

The covariance matrix for r is given by

$$K_r = E[rr^\dagger] = \Gamma^\dagger \Sigma \Gamma + N_0 I_N \, . \tag{2.7}$$

3. UNCONSTRAINED ML ESTIMATOR

In this section, we introduce the ML estimator for the spectral matrix Σ defined in the model (2.6). From (2.7), the likelihood function for Σ is

$$L(r,\Sigma) = -\ln \det (\Gamma^\dagger \Sigma \Gamma + N_0 I_N) - r^\dagger (\Gamma^\dagger \Sigma \Gamma + N_0 I_N)^{-1} r \, . \tag{3.1}$$

Maximizing the likelihood with respect to Σ yields the necessary trace condition which a positive definite estimate $\hat{\Sigma}$ must satisfy [2]:

$$Tr \; [\Gamma \hat{K}_r^{-1} (rr^\dagger - \hat{K}_r) \hat{K}_r^{-1} \Gamma^\dagger \delta \hat{\Sigma}] = 0 \, , \tag{3.2}$$

with

$$\hat{K}_r = \Gamma^\dagger \hat{\Sigma} \Gamma + N_0 I_N \, ,$$

for all admissible $M \times M$ diagonal matrices $\delta \hat{\Sigma}$ [2]. This trace condition is a nonlinear equation in $\hat{\Sigma}$. Generally it cannot be solved directly in closed-form, so some numerical search procedure must be implemented. An approach is the expectation-maximization (EM) algorithm of Dempster *et al.* [3] used in [2]. The complete data are defined as (c, w). An initial estimate $\hat{\Sigma}^{(0)}$ is selected. At step k+1 (k = 0,1,..) the estimate is updated according to

$$\hat{\Sigma}^{(k+1)} = \underset{\Sigma}{argmax} \; Q(\Sigma | \hat{\Sigma}^{(k)}) \tag{3.3}$$

where

$$Q(\Sigma | \hat{\Sigma}^{(k)}) = -\sum_{i=0}^{M-1} \ln \sigma^2(i) - \sum_{i=0}^{M-1} \frac{E[\,|c(i)|^2 \,|\, r, \hat{\Sigma}^{(k)}]}{\sigma^2(i)} \, , \tag{3.4}$$

and

$$E[\,|c(i)|^2 \,|\, r, \hat{\Sigma}^{(k)}] = [\hat{\Sigma}^{(k)} - \hat{\Sigma}^{(k)} \Gamma K_r^{-1\,(k)} \Gamma^\dagger \hat{\Sigma}^{(k)} + \hat{\Sigma}^{(k)} \Gamma K_r^{-1\,(k)} rr^\dagger K_r^{-1\,(k)} \Gamma^\dagger \hat{\Sigma}^{(k)}]\,(i,i) \, , \tag{3.5}$$

where

$$K_r^{(k)} = \Gamma^\dagger \hat{\Sigma}^{(k)} \Gamma + N_0 I_N \, .$$

This algorithm produces a sequence of estimates

$$\hat{\sigma}^2(i)^{(k+1)} = E[\,|c(i)|^2 \,|\, r, \hat{\Sigma}^{(k)}] \tag{3.6}$$

having increasing likelihood. It can be shown that the stable points of this algorithm satisfy the necessary trace condition (3.2) for a maximizer [2].

The performance of this estimator has been evaluated in previous studies and compared to traditional spectrum estimators such as the periodogram [4, 5]. It has been shown that

(1) the ML estimates have a very low bias;

(2) the ML estimates offer a better trade-off between variance and resolution than the periodogram;

(3) the ML estimates are not mean-square consistent. This unfortunate result is due to the fact that the dimension M of the parameter space is at least equal to the number of data N. We are actually dealing with a *small-sample* ML estimation problem. This observation is one factor that motivated the study presented in the next sections.

4. REGULARIZATION OF THE ML ESTIMATES

According to Hadamard's classical definition, a problem is said to be well posed if: (1) it has a solution, (2) the solution is unique, and (3) the solution varies continuously with the data. A problem is ill posed if it is not well posed. Small changes in the data can then produce unbounded changes in the estimates. Typical examples are inversion of certain Fredholm integral equations and the reconstruction of functions from truncated Fourier transforms [6].

The ML estimation procedure produces similar artifacts when the number of parameters (M) is large compared to the number of data samples (N). The estimates are extremely sensitive to small changes in the data and exhibit a very rough shape characterized by very sharp peaks and low valleys. We can say that the problem is *practically,* if not *technically,* ill posed. Such a behavior is highly undesirable, and a *regularization* of the estimates is required. This concept was introduced by Tikhonov [7]. A large bulk of mathematical literature has been written on this subject; see for instance Bertero [8] for a tutorial.

One possible method for the regularization of estimates is the method of sieves, introduced by Grenander [1]. According to Grenander's definition, a *sieve in a parameter space A* is a family of subsets $S(\mu)$ of A indexed by a positive parameter μ called the *mesh size.* A restricted ML estimate exists over each set $S(\mu)$. As the mesh size tends to zero the sets $S(\mu)$ will be large enough to allow the ML solution to converge to any solution in A [1, p.357]. For the problem at hand, the choice of a "good" value of μ will depend on the data record size, the noise level, and possibly other factors. A major problem is to find sieves which will make the ML estimates converge and possess some desirable practical features such as analytical and/or computational tractability. For instance, one might consider a *convolution sieve:*

$$S(\mu) = \left\{ \sigma^2(i) \mid \sigma^2(i) = \eta(i) * \psi_\mu(i) \;\; 0 \le i \le M-1 \right\} . \tag{4.1}$$

where $\sigma^2(i)$ is the constrained spectrum, $\eta(i)$ is any spectrum, and $\psi_\mu(i)$ is a convolution kernel, or "window", with mesh equal to μ. However, it is not clear that finding spectral estimates within this sieve is computationally tractable. In 1985 Chow and Grenander recognized that regularizing the ML spectrum estimates with a convolution sieve appears to be a *formidable* task, so they studied other types of estimators [9].

Here we recommend a regularization method based on nonorthogonal expansions of the unknown density function. This approach leads to computationally tractable ML estimates. As discussed in §5.3, we also expect the estimates to be asymptotically consistent.

The regularization procedure developed in the next section will be applied to the technically ill-posed spectrum estimation problem in which the ratio M/N tends to infinity, where the approximate model (2.4) converged to the exact model (2.1). We will show how to solve this infinite-dimensional parameter estimation problem.

5. A METHOD OF SIEVES BASED ON NONORTHOGONAL EXPANSIONS

The spectrum $S(f)$ belongs to a Hilbert space of functions defined over $[-f_{\max}, f_{\max}]$. Denote by $\mathbf{H}$ this Hilbert space, and by $\mathbf{H}_+$ the subset of all nonnegative functions in $\mathbf{H}$. Typically $\mathbf{H}$ is $L_2[-f_{\max}, f_{\max}]$, the space of square-integrable functions over $[-f_{\max}, f_{\max}]$. Our goal is to represent the spectral function in $\mathbf{H}_+$ in terms of a set of basis functions having localized support. This permits a good *local* representation of a function. As such, this is related to the concept of wavelets.

5.1. Definition of the sieve

Consider a spectral M-vector $\underline{\sigma}^2$. To this vector we associate a step function $S(f)$ according to the map ST_M defined as follows:

$$ST_M[\underline{\sigma}^2] \, : \, \mathbf{R}^M \to \mathbf{H} \, : \, \underline{\sigma}^2 \to S(f) = ST_M[\underline{\sigma}^2] = \sum_{i=0}^{M-1} \sigma^2(i) \chi_i(f) , \tag{5.1}$$

where $\{f_i\}_{i=0}^{M-1}$ is the same uniform partition of the interval $[-f_{\max}, f_{\max}]$ as in Section 2, $\mathbf{R}^M$ is the M-dimensional euclidean space, and $\chi_i(f)$ is the indicator function over the i-th partition interval.

Let $\{\psi_m(f)\}_{m=0}^{\infty}$ be a nonorthogonal set of positive basis functions in $\mathbf{H}_+$. Consider the set $\mathbf{A}$ of all functions in the span of the $\{\psi_m\}_{m=0}^{\infty}$, with positive coefficients:

$$\mathbf{A} = \left\{ S(f) \mid S(f) = \sum_{m=0}^{\infty} a(m)\psi_m(f) \quad , a(m) > 0 \right\} . \tag{5.2}$$

We define our parameter set to be $\mathbf{A}$. Clearly $\mathbf{A}$ is a subset of $\mathbf{H}_+$. It would be desirable to have $\{\psi_m\}_{m=0}^{\infty}$ be designed so that $\mathbf{A}$ is dense in $\mathbf{H}_+$. It is well known that this can be achieved with $\{\psi_m\}_{m=0}^{\infty}$ defined as indicator functions over a partition of the frequency domain, in the limit as the partition size tends to zero. How $\mathbf{A}$ is made dense in $\mathbf{H}_+$ for more elaborate designs remains to be investigated.

Now define the following step-function approximation of $\psi_m(f)$:

$$\psi_{M,m}(f) = ST_M\,[\underline{\psi}_{M,m}] , \tag{5.3}$$

where $\underline{\psi}_{M,m}$ is a M-vector of samples of $\psi_m(f)$ taken on the partition of the frequency interval. As $M \to \infty$, $\psi_{M,m}(f)$ converges to $\psi_m(f)$ in the norm of $\mathbf{H}$. We define a sieve $\mathrm{S}(M, Q)$ on $S(f)$ in $\mathbf{A}$ by truncating the series representation (5.2) to a finite number of terms Q and considering M-step approximations to the basis functions,

$$\mathrm{S}(M, Q) = \left\{ S(f) \mid S(f) = \sum_{m=0}^{Q-1} a(m)\psi_{M,m}(f) \quad , a(m) > 0 \right\} . \tag{5.4}$$

The functions in the set $\mathrm{S}(M, Q)$ are constrained to be M-step functions of the type (5.1). For the time being, we keep M finite, but ultimately M will be allowed to tend to infinity. We view $\{a(m)\}_{m=0}^{Q-1}$ as being Q unknown parameters for which ML estimates are sought.

Why nonorthogonal basis functions?

Before considering the general problem, we mention the special case where the basis functions considered in (5.4) are Q indicator functions over disjoint intervals covering the frequency domain. In this instance, the computation of the $a(m)$'s is straightforward [*]. However these basis functions have the major disadvantage that the estimates are step functions, and so display discontinuities that are uncharacteristic of spectral densities.

To achieve good representations for a richer class of functions, we have to forgo the aforementioned restriction on the structure of the basis functions. A possible design of the basis functions might be based upon *dilations* and *translations* of a smooth basic function. Such a design is used in the wavelet literature because it provides a convenient *multiscale* representation of the function of interest [10, 11]. This is usually achieved more easily with nonorthogonal basis functions [11].

Our strategy in using other basis functions will have practical interest if a tractable parameter estimation procedure can be derived. This issue is addressed in the next section.

5.2 Computation of the sieve estimator

To each function $S(f)$ in the constrained set (5.4) is associated a spectral M-vector $\underline{\sigma}^2$ via the inverse map

$$\underline{\sigma}^2 = ST_M^{-1}\,[S(f)] . \tag{5.5}$$

---

[*] The maximization problem (5.7) can then be solved in closed form.

Since the map ST_M^{-1} is linear, we conclude from (5.1), (5.4) and (5.5) that the vector $\underline{\sigma}^2$ is constrained as follows.

$$\underline{\sigma}^2 = \sum_{m=0}^{Q-1} a(m)\underline{\psi}_{M,m} . \tag{5.6}$$

The ML estimates of the $a(m)$'s can be obtained by using the EM algorithm described in Section 3. At each step the expression (3.4) is maximized with respect to $\hat{\sigma}^2$ subject to the sieve constraint (5.5). This amounts to finding the maximum with respect to a of

$$Q(a|\hat{a}^{(k)}) = -\sum_{i=0}^{M-1} \ln\left[\sum_{m=0}^{Q-1} a(m)\underline{\psi}_{M,m}(i)\right] - \sum_{i=0}^{M-1} \frac{E[|c(i)|^2|r,\hat{a}^{(k)}]}{\sum_{m=0}^{Q-1} a(m)\underline{\psi}_{M,m}(i)} . \tag{5.7}$$

This maximization problem is very difficult and must be solved via some numerical procedure. Keeping in mind that this operation has to be repeated at every iteration of the EM-algorithm, one is hardly attracted by this prospect. We propose the following method instead.

An EM algorithm has been developed, based on the definition of a new complete data space for producing the estimates. This method is set up as an unconstrained maximization problem in a Q-dimensional parameter space and, as such, it does not involve any computation in **H**. Because of the existence of such an algorithm, our estimation procedure based on nonorthogonal expansions turns out to be computationally feasible. This usually represents an impressive computational saving and is a major attractive feature of this approach.

Complete/Incomplete Data Spaces

Write c as the sum of Q independent vectors:

$$c = \sum_{m=0}^{Q-1} \tilde{c}_m , \tag{5.8}$$

where $\tilde{c}_m$ is a M-vector, sample of a 0-mean gaussian process with diagonal covariance $a(m)\tilde{\Psi}_m$, with $\tilde{\Psi}_m$ a diagonal matrix made of $\{\ \underline{\psi}_{M,m}(i), i \in [0\,..M-1]\ \}$. These Q processes are independent. Now define the J_m-vector c_m as the restriction of $\tilde{c}_m$ to its support set S_m. The covariance Ψ_m of c_m is made of the nonzero entries of $\tilde{\Psi}_m$. With these notations our model (2.6) becomes

$$\begin{aligned} r &= \sum_{m=0}^{Q-1} \Gamma^{\dagger}\tilde{c}_m + w \\ &= \sum_{m=0}^{Q-1} \Gamma_m^{\dagger} c_m + w . \end{aligned} \tag{5.9}$$

We define the complete data as ($\{c_m\}_{m=0}^{Q-1}$; w). Following this definition we write the loglikelihood for the complete data as

$$\begin{aligned} l_{cd}(a) &= \sum_{m=0}^{Q-1} \ln p(c_m:a) \\ &= -\sum_{m=0}^{Q-1} \ln\det(a(m)\Psi_m) - \sum_{m=0}^{Q-1} c_m^{\dagger}(a(m)\Psi_m)^{-1} c_m . \end{aligned}$$

Discarding all terms not involving a, we obtain:

$$l_{cd}(a) = -\sum_{m=0}^{Q-1} J_m \ln a(m) - \sum_{m=0}^{Q-1} \frac{1}{a(m)} \sum_{i \in S_m} \frac{|c_m(i)|^2}{\underline{\psi}_{M,m}(i)} . \tag{5.10}$$

The conditional expectation of $l_{cd}(a)$ is then

$$Q(a|\hat{a}^{(k)}) = -\sum_{m=0}^{Q-1} J_m \ln a(m) - \sum_{m=0}^{Q-1} \frac{1}{a(m)} \sum_{i \in S_m} \frac{E\,[|c_m(i)|^2|r,\hat{a}^{(k)}]}{\underline{\psi}_{M,m}(i)} . \tag{5.11}$$

Taking the partial derivative of (5.11) with respect to $a(m)$, and setting the result to zero yields

$$\hat{a}(m)^{(k+1)} = \frac{1}{J_m} \sum_{i \in S_m} \frac{E\,[|c_m(i)|^2 | r, \hat{a}^{(k)}]}{\underline{\psi}_{M,m}(i)} . \tag{5.12}$$

Next we evaluate the conditional expectation of $|c_m(i)|^2$ in (5.12). Define

$$\hat{c}_m^{(k)} = E\,[\, c_m \,|\, r, \hat{a}^{(k)}] . \tag{5.13}$$

The conditional expectation in (5.12) is the i-th diagonal element of the matrix

$$\begin{aligned} E\,[\, c_m c_m^{\dagger} | r, \hat{a}^{(k)}] &= E\,[\, (c_m - \hat{c}_m^{(k)})\,(c_m - \hat{c}_m^{(k)})^{\dagger} | r, \hat{a}^{(k)}] + \hat{c}_m^{(k)} \hat{c}_m^{(k)\dagger} \\ &= (K_{c_m r}^{(k)} K_r^{-1\,(k)} r)(K_{c_m r}^{(k)} K_r^{-1\,(k)} r)^{\dagger} + K_{c_m c_m}^{(k)} - K_{c_m r}^{(k)} K_r^{-1\,(k)} K_{r c_m}^{(k)} , \end{aligned} \tag{5.14}$$

where we denote by K_{xy} the conditional correlation $E\,[xy^{\dagger}|\hat{a}^{(k)}]$ of two random vectors x and y. Now, the expectations are evaluated from the model (5.9). Note that because the c_m's are independent, the conditional expectations on $\hat{a}^{(k)}$ are simply conditional expectations on $\hat{a}(m)^{(k)}$. We get

$$K_{c_m r}^{(k)} = E\,[\, c_m r^{\dagger} | \hat{a}^{(k)}] = \hat{a}(m)^{(k)} \Psi_m \Gamma_m , \tag{5.15a}$$

$$K_r^{(k)} = E\,[\, r r^{\dagger} | \hat{a}^{(k)}] = \sum_{j=0}^{Q-1} \hat{a}(j)^{(k)} \Gamma_j^{\dagger} \Psi_j \Gamma_j + N_0 I_N , \tag{5.15b}$$

$$K_{c_m c_m}^{(k)} = E\,[\, c_m c_m^{\dagger} | \hat{a}^{(k)}] = \hat{a}(m)^{(k)} \Psi_m . \tag{5.15c}$$

Substituting the expressions (5.15a,c) into (5.14) yields

$$E\,[\, c_m c_m^{\dagger} | r, \hat{a}^{(k)}] = \hat{a}(m)^{(k)} \Psi_m + \hat{a}(m)^{(k)} \Psi_m \Gamma_m K_r^{-1\,(k)} [\, r r^{\dagger} - K_r^{(k)}]\, K_r^{-1\,(k)} \Gamma_m^{\dagger} \Psi_m \hat{a}(m)^{(k)} , \tag{5.16}$$

with $K_r^{(k)}$ given in (5.15b). Taking the i-th diagonal element of (5.16) and substituting in (5.12), we get

$$\begin{aligned} \hat{a}(m)^{(k+1)} &= \frac{1}{J_m} \sum_{i \in S_m} \frac{1}{\underline{\psi}_{M,m}(i)} \Big[\, \hat{a}(m)^{(k)} \underline{\psi}_{M,m}(i) + \hat{a}(m)^{(k)} \underline{\psi}_{M,m}(i) \\ &\qquad \times [\Gamma_m K_r^{-1\,(k)} (r r^{\dagger} - K_r^{(k)}) K_r^{-1\,(k)} \Gamma_m^{\dagger}]_{ii}\, \underline{\psi}_{M,m}(i) \hat{a}(m)^{(k)} \Big] . \\ &= \hat{a}(m)^{(k)} + \hat{a}(m)^{(k)} \Big[\frac{1}{J_m} \sum_{i \in S_m} \underline{\psi}_{M,m}(i) \\ &\qquad \times [\Gamma_m K_r^{-1\,(k)} (r r^{\dagger} - K_r^{(k)}) K_r^{-1\,(k)} \Gamma_m^{\dagger}]_{ii} \Big]\, \hat{a}(m)^{(k)} . \end{aligned} \tag{5.17}$$

Now defining the $N \times N$ matrices

$$Z^{(k)} = K_r^{-1\,(k)} (r r^{\dagger} - K_r^{(k)}) K_r^{-1\,(k)} \tag{5.18}$$

and

$$K_m = \frac{1}{J_m} \Gamma_m^{\dagger} \Psi_m \Gamma_m , \tag{5.19}$$

the term between brackets in the right-hand side of (5.17) can be written

$$\begin{aligned} \frac{1}{J_m} \sum_{i \in S_m} \underline{\psi}_{M,m}(i)\, (\Gamma_m Z^{(k)} \Gamma_m^{\dagger})_{ii} &= \frac{1}{J_m} Tr \Big[\, \Psi_m \Gamma_m Z^{(k)} \Gamma_m^{\dagger} \Big] \\ &= Tr \Big[\, \frac{1}{J_m} \Gamma_m^{\dagger} \Psi_m \Gamma_m Z^{(k)} \Big] \\ &= Tr \Big[\, K_m Z^{(k)} \Big] . \end{aligned} \tag{5.20}$$

Substituting (5.20) in (5.17) we get the update equations at stage k of the algorithm:

$$\hat{a}(m)^{(k+1)} = \hat{a}(m)^{(k)} + \hat{a}(m)^{(k)}\, Tr \Big[\, K_m Z^{(k)} \Big]\, \hat{a}(m)^{(k)} , \tag{5.21}$$

$$K_r^{(k+1)} = \sum_{j=0}^{Q-1} \hat{a}(j)^{(k)} K_j + N_0 I_N \,. \tag{5.22}$$

Comments

(1) The update equations (5.21) and (5.22) highlight the role played by the basis covariances K_m in this representation.

(2) As $M \to \infty$, the algorithm for estimating the coefficients $a(m)$ is well-behaved. The reason is that the basis covariance K_m in (5.21) is then given by the limit of the matrix product (5.19) as $M \to \infty$, which is a matrix of integrals with entry (n, m) given by

$$K_m(n, m) = \int_{-f_{max}}^{f_{max}} \exp(-j2\pi f(n-m)\Delta t)\, \psi_m(f)\, df \,.$$

We can thus let $M \to \infty$ without any loss in stability of the estimates or any increse in computations.

(3) Implementation of the solution:

Each matrix K_m is constant and can be computed off-line. At each step the covariance matrix K_r in (5.22) must first be inverted, which accounts for N^3 operations. Following this, updating each $\hat{a}(m)$ just requires N^2 multiplications/additions. The complexity of each step of the EM algorithm is then $N^3 + QN^2$, as compared to $N^3 + MN^2$ in the unconstrained case. The saving is impressive for infinite M, keeping $Q \sim N$.

(4) It is easily shown that the trace appearing in the update equation (5.21) is the partial derivative of the incomplete-data loglikelihood with respect to $a(m)$.

5.3. Some Regularization Aspects

The most attractive feature of this approach is a theoretical one. We are effectively estimating Q parameters instead of M as in the unconstrained problem. The choice of Q is somewhat arbitrary and depends on the way we have decided to represent the spectral function. As a rule of thumb it seems desirable to have Q (# of parameters) no larger than N (# of data). This can be done while letting M be infinite, a result that could not have been obtained without regularization of the ML estimates. The process in our model then approaches the limiting case of a (nonperiodic) stationary process, and as such does not involve any model approximation.

Consistency of the estimates

A major goal when defining a sieve is to ensure consistency of the estimates. Although no thorough study has yet been undertaken for the sieve (5.4), we believe that consistency can be achieved, provided that Q tends to infinity at a slower rate than N. Typically : $Q \sim N^{\alpha}$, with $0 < \alpha < 1$. Another objective is to ensure rapid convergence of the estimates. Clearly the issue here is the design of the basis functions, which is one of our current research areas.

CONCLUSIONS

We have proposed a method for a estimating a spectral function represented by a linear combination of basis functions. The task of finding the ML estimates of the coefficients of these basis functions looks formidable at first (refer to the optimization problem (5.7)). However we propose a method based on the decomposition (5.8) of the spectral process into a sum of Q independent spectral processes with known covariance, up to a scale factor. The subsequent reformulation of the complete/incomplete data spaces leads to the derivation of a powerful algorithm for evaluating these scale factors.

The main features of this approach are: (1) *flexibility*. The choice of basis functions is wide open. In particular we can choose the shape of the basis functions, amount of overlap, and support set extent (possibly non-uniform). (2) *using an exact model*. No approximation of the exact model for the stochastic process is needed. (3) *computational efficiency*. The computational complexity of the algorithm is a function of the dimension of the parameter space, not of the partition size of the frequency axis. An important result is that it is now possible to let $M \to \infty$ without any increase in the number of computations.

Several interesting questions remain open issues. In particular it is desired to achieve mean-square consistency of the estimates. As mentioned in §5.2.2, this might be done by letting the sieve grow at an appropriate rate as $N \to \infty$. The issue of uniqueness of the estimates should also be investigated. Finally, the design of the basis functions will determine the overall estimator performance, as far as convergence rate and sensitivity to noise are concerned.

Also note that the setup of this problem is very general and can be applied without major difficulties to the estimation of higher-dimensional spectral functions. In particular, the spectra of interest in the radar-imaging problem described in [2] are two-dimensional. The application of the ideas in this paper to the radar-imaging problem is another area of active research.

REFERENCES

[1] U. Grenander, *Abstract Inference*, Wiley 1981.

[2] D. L. Snyder, J. A. O'Sullivan and M. I. Miller, "The Use of Maximum-Likelihood Estimation for Forming Images of Diffuse Radar-Targets from Delay-Doppler Data", Vol. 35, No. 3, *IEEE Trans. on Information Theory*, pp. 536-548, 1989.

[3] A.D. Dempster, N.M. Laird and D.B. Rubin, "Maximum-Likelihood from Incomplete Data via the EM Algorithm", *J. Royal Stat. Soc., Vol. B39*, pp. 1-37, 1977.

[4] P. Moulin, "Maximum-Likelihood Spectrum Estimation of Periodic Processes from Noisy Data", *Washington University, ESSRL internal report.*

[5] P. Moulin, D.L. Snyder and J.A. O'Sullivan, "Maximum-Likelihood Spectrum Estimation of Periodic Processes from Noisy Data", *Proc. of the 1989 Conference on Information and Systems Science*, Johns Hopkins University, Baltimore, 1989.

[6] L.S. Joyce and W.L. Root, "Precision Bounds in Superresolution Processing", *J. Opt. Soc. Am., Vol.1*, pp. 149-168, 1984.

[7] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*, Winston & Sons, Washington D.C. 1977.

[8] M. Bertero, C. De Mol, G. Viano, "The Stability of Inverse Problems", Chapter 5 in *Inverse Scattering Problems*, Springer-Verlag 1980.

[9] Y. Chow and U. Grenander, "A Sieve Method for the Spectral Density", *Ann. Stat., Vol. 13, No. 3*, pp. 998-1010, 1985.

[10] M. Frasier and B. Jawerth, "Decomposition of Besov Spaces", *Indiana Univ. Math. Journal, Vol. 34, No. 3*, 1985, pp.777-799.

[11] I. Daubechies, A. Grossmann and Y. Meyer, "Painless Nonorthogonal Expansions", *J. Math. Phys., Vol. 27, No. 5*, 1986.

IDENTIFIABILITY OF AR PARAMETERS OF NONCAUSAL ARMA SIGNALS FROM OUTPUT CUMULANTS

JITENDRA K. TUGNAIT

Dept. of Electrical Engineering, Auburn University, Alabama 36849

ABSTRACT

The problem of identifiability of the autoregressive (AR) parameters of a stable, scalar, noncausal ARMA(p,q) signal model from the output cumulants is considered. The system is driven by an i.i.d. non-Gaussian sequence that is not observed. We first derive two Yule-Walker-type cumulant recursions dependent upon the system poles. Then we associate two infinite block-Hankel matrices with these recursions and derive lower and upper bounds on the rank of the Hankel matrices in terms of the number of system causal (inside the unit disk) and anticausal (outside the unit disk) poles. These results are then used to prove the identifiability of the AR parameters of the noncausal ARMA model from the system output cumulants. Estimation of the AR parameters requires a nonlinear solution unlike the causal (anticausal) case where a linear solution suffices.

I. INTRODUCTION

Model Assumptions

Let y(t) denote a p-th order stationary scalar ARMA (autoregressive moving average) signal given by

$$\sum_{i=0}^{p} a_i\, y(t-i) = \sum_{i=0}^{q} b_i\, w(t-i) \tag{1.1}$$

where t is an integer (discrete-time). The following conditions are assumed to hold.

(H1) $A(z) = \sum_{i=0}^{p} a_i z^{-i} \neq 0$ for $|z|=1$ where z is a complex variable.

(H2) The random sequence $\{w(t)\}$ is zero-mean, i.i.d., and non-Gaussian with $E\{w^2(t)\} =: \sigma_w^2 \neq 0$, $E\{w^3(t)\} =: \gamma_{3w} \neq 0$, and $E\{w^6(t)\} < \infty$.

(H3) There are no pole-zero cancellations in (1.1), and $a_0 \neq 0$, $a_p \neq 0$, $b_0 \neq 0$ and $b_q \neq 0$.

As in [6] we may factor the polynomial A(z) into two polynomials, one having all its roots inside the unit circle, the other with its roots outside the unit circle. That is, following [6], we have

$$A(z) = d(z) A_1(z) A_2(z) \tag{1.2}$$

where

$$A_1(z) = 1 + \sum_{i=1}^{p_1} a_{1i} z^{-i} = \prod_{i=1}^{n_c} (1-\alpha_{ci} z^{-1})^{n_{ci}}, \quad |\alpha_{ci}|<1, \ \ \alpha_{ci} \neq \alpha_{cj} \ \text{ for } \ i \neq j \tag{1.3}$$

$$A_2(z^{-1}) = 1 + \sum_{i=1}^{p_2} a_{2i} z^{-i} = \prod_{i=1}^{n_{ac}} (1-\alpha_{aci}\, z^{-1})^{n_{aci}}, \quad |\alpha_{aci}|>1, \;\; \alpha_{aci}\neq\alpha_{acj} \;\; \text{for } i\neq j \tag{1.4}$$

$$d(z) = d_0 z^{-p_2} \;\; \text{for some real } d_0 \tag{1.5}$$

$$p = p_1 + p_2, \;\; p_1 = \sum_{i=1}^{n_c} n_{ci}, \;\; p_2 = \sum_{i=1}^{n_{ac}} n_{aci}. \tag{1.6}$$

Thus, there are p_1 system poles ("causal poles") that lie inside the unit circle, and p_2 poles ("anticausal poles") that lie outside the unit circle. In the sequel, we will take $d_0{:=}1$ without any loss of generality by "absorbing" it in $\{w(t)\}$.

Finally we impose the restriction of the knowledge of the system order.

(H4) The system orders, namely, the integers q, p_1 and p_2 are known.

Motivation

Models such as (1.1) arise in several areas of considerable practical interest such as geophysical signal processing with chirped acoustic source (vibrator), image processing with noncausal blurs, etc.; see [3,[4],[6] and [7], and references therein. System identification and parameter estimation problems for noncausal AR (autoregressive) models using both second- and higher- order statistics has been addressed in [3],[4] and [5] (among others). When measurement noise is *colored* Gaussian, the approaches of [3],[4] and [5] will yield biased parameter estimates. In such a case it is of interest to use only the higher-order statistics since the higher-order cumulants of Guassian processes are identically zero.

This paper addresses the fundamental issue of identifiability of the AR parameters of a noncausal ARMA model (that is exponentially stable however) from the third-order cumulants of the model output. Here we extend the results of [2] to noncausal models. One cannot derive a Yule-Walker like recursion for the output cumulants in the time-domain for noncausal models as can be done for causal models; see [2] and [10]. Therefore, a somewhat different approach has to be taken.

We will use the third- order statistics of the signal observations to estimate the AR parameters as in [2]. For a zero-mean stationary process $\{y(t)\}$ the third-order cumulant function may be defined as

$$C_3(t_1,t_2) = E\{y(t)y(t+t_1)y(t+t_2)\}. \tag{1.7}$$

II. YULE-WALKER-TYPE CUMULANT RECURSIONS

We examine the third-order "1-D" spectrum, defined as

$$S_3(z;m) := \sum_{\tau=-\infty}^{\infty} C_3(\tau+m,\tau)\, z^{-\tau} = \gamma_{3w}\, H_2(z;m)\, H(z^{-1}), \tag{2.1}$$

for the model (1) under the assumptions (H1)-(H3) where H(z) is the transfer function of (1)

$$H(z) := [\sum_{i=0}^{q} b_i z^{-i}]/[\sum_{i=0}^{p} a_i z^{-i}] \tag{2.2}$$

and

$$H_2(z;m) := [z^m H(z)] * H(z)$$

(the symbol * denotes convolution).

We may rewrite (2.2) as

$$H(z) = [\sum_{j=0}^{q-p} c_j z^{-j}]\, u_{q-p} + [\sum_{i=1}^{p} \tilde{b}_i z^i] / [z^p + \sum_{i=1}^{p} a_i z^{p-i}] \tag{2.3}$$

where u_i is the unit step function (0 for $i<0$ and 1 otherwise).

By a partial fraction expansion of H(z) and its inverse Z-transform we obtain

$$h(i) = [\sum_{j=0}^{q-p} c_j\, \delta(i-j)]\, u_{q-p} + \bar{h}(i) \tag{2.4}$$

where

$$\bar{h}(i) = \sum_{j=1}^{n_c} \sum_{k=1}^{n_{cj}} d_{jk}\, e_{ki}\, \alpha_{cj}^{i}\, u_i + \sum_{j=1}^{n_{ac}} \sum_{k=1}^{n_{acj}} \bar{d}_{jk}\, e_{ki}\, \alpha_{acj}^{i}\, u_{-(i+1)}\ . \tag{2.5}$$

$$e_{ki} := \begin{cases} (i+1)(i+2)\cdots(i+k-1)[(k-1)!]^{-1} & \text{for } k>1 \\ 1 & \text{for } k=1 \end{cases} \tag{2.6}$$

and d_{jk} and $\bar{d}_{jk}$ are some constants of expansion.

We now wish to look at the product h(i)h(i+m) for a fixed m, and the Z-transform of the sequence {h(i)h(i+m)}, $H_2(z;m)$. From the Appendix, we have

$$H_2(z;m) = \sum_{j=1}^{L_c} \sum_{k=1}^{L_{cj}} r_{jk}(m)\,(1-\beta_{cj} z^{-1})^{-k} + \sum_{j=1}^{L_{ac}} \sum_{k=1}^{L_{acj}} \bar{r}_{jk}(m)\,(1-\beta_{acj} z^{-1})^{-k} z^m$$
$$+ [\sum_{i=1}^{m} c_{1i} z^i]\, u_{m-1} + [\sum_{i=p-q}^{m} c_{2i} z^i]\, u_{q-p}\ , \qquad (m \geq 0). \tag{2.7}$$

After some algebraic manipulations we have

$$H_2(z;m) = N_2(z) / [A_{2c}(z)\, A_{2ac}(z)] \qquad (m \geq 0) \tag{2.8}$$

where

$$A_{2c}(z) := \prod_{j=1}^{L_c} (z-\beta_{cj})^{L_{cj}} = \sum_{i=0}^{\bar{p}_1} a_{2ci} z^i \neq 0 \text{ for } |z| \geq 1 \tag{2.9}$$

$$A_{2ac}(z) := \prod_{j=1}^{L_{ac}} (z-\beta_{acj})^{L_{acj}} = \sum_{i=0}^{\bar{p}_2} a_{2aci} z^i \neq 0 \text{ for } |z| \leq 1 \tag{2.10}$$

$$\bar{p}_1 := 0.5p_1(p_1+1) = \sum_{j=1}^{L_c} L_{cj} \; ; \quad \bar{p}_2 := 0.5p_2(p_2+1) = \sum_{j=1}^{L_{ac}} L_{acj} \tag{2.11}$$

$$N_2(z) := \sum_{i=i_{20}}^{i_{21}} d_{2i}\, z^i \; ; \quad i_{21} = \bar{p}_1+\bar{p}_2+m \; ; \quad i_{20} = (p-q)\, u_{q-p} \; . \tag{2.12}$$

Finally, it is easy to show that for any integer m

$$H_2(z;m) = z^m\, H_2(z;-m)\, . \tag{2.13}$$

By (2.2) and (2.8)-(2.13) we have the product

$$H_2(z;m)H(z^{-1}) = N_3(z)/[\,A_{3ac}(z^{-1})\, A_{3c}(z)\,] \text{ for any } m \tag{2.14}$$

where

$$A_{3ac}(z^{-1}) := \prod_{j=1}^{L_{ac}} (1-\beta_{acj}^{-1}\, z)^{L_{acj}} \prod_{i=1}^{n_c} (1-\alpha_{ci}\, z)^{n_{ci}} = \sum_{i=0}^{p_1+\bar{p}_2} a_{ci}\, z^i \neq 0 \text{ for } |z|\leq 1 \tag{2.15}$$

$$A_{3c}(z) := \prod_{j=1}^{L_c} (1-\beta_{cj}\, z^{-1})^{L_{cj}} \prod_{i=1}^{n_{ac}} (1-\alpha_{aci}^{-1}\, z^{-1})^{n_{aci}} = \sum_{i=0}^{p_2+\bar{p}_1} a_{aci}\, z^{-i} \neq 0 \text{ for } |z|\geq 1 \tag{2.16}$$

$$N_3(z) := \sum_{i=i_{30}}^{i_{31}} d_{3i}\, z^i \; ; \tag{2.17}$$

$$i_{31} = q+\bar{p}_2-p_2+|m|+m\, u_{-m} \; ; \quad i_{30} = -p_2-\bar{p}_1-(q-p)\, u_{q-p}+m\, u_{-m} \; . \tag{2.18}$$

Now use (2.1) and (2.14) to obtain

$$\sum_{i=0}^{p_1+\bar{p}_2} a_{ci}\, C_3(i-\tau+m, i-\tau) = \gamma_{3w} \sum_{i=i_{30}}^{i_{31}} d_{3i}\, \tilde{h}(i-\tau) \tag{2.19}$$

where $\tilde{h}(i)$ is the inverse Z-transform of the causal transfer function $A_{3c}^{-1}(z)$; hence, $\tilde{h}(i)=0$ for $i<0$. Use this fact in (2.19) to conclude that

$$\sum_{i=0}^{p_1+\bar{p}_2} a_{ci}\, C_3(i-\tau+m, i-\tau) = 0 \text{ for } \tau\geq i_{31}+1.$$

Finally, note from (2.15) that $a_{c0}=1$ so that we have

$$C_3(-\tau+m,-\tau) = C_3(\tau,m) = -\sum_{i=1}^{p_1+\bar{p}_2} a_{ci}\, C_3(\tau-i,m) \text{ for } \tau\geq i_{31}+1 \text{ and any } m. \tag{2.20}$$

By mimicking the above development (consider a "time-reversal" of y(t) to derive the above results for (1.1), and then do another "time-reversal"), we can also show that recursion (2.21) is governed by the "remaining" poles of (2.1):

$$C_3(\tau-m,\tau) = C_3(-\tau,-m) = -\sum_{i=1}^{p_2+\bar{p}_1} a_{aci}\, C_3(-\tau+i,-m) \text{ for } \tau\geq i'_{31}+1 \text{ and any } m, \tag{2.21}$$

where

$$i'_{31} = q+\bar{p}_1-p_1+|m|+m\,u_{-m}, \tag{2.22}$$

and a_{aci}'s are as defined in (2.16).

Eqns. (2.20) and (2.21) are the two basic cumulant recursions for a general NCARMA(p_1,p_2,q) model. We emphasize that *minimality* of the order of these two recursions is not guaranteed. If there is any pole-zero cancellation in (2.14) such that degree of $A_{3ac}(z)$ is reduced, then there is a corresponding change in the order (and the coefficients) of the recursion (2.20).

III. MINIMAL ORDER OF THE CUMULANT RECURSIONS

As noted in the previous section, the cumulant recursions (2.20) and (2.21) are not necessarily minimal, i.e., the output cumulants of the model (1.1) could obey recursions like (2.20) and (2.21) with orders smaller than $p_1+\bar{p}_2$ and $p_2+\bar{p}_1$, respectively, owing to pole-zero cancellations in (2.14). As in [2], we show that by stacking up 1-D scalar recursions into a vector recursion, certain minimal order can be guaranteed. We now give lower and upper bounds to these orders, and more importantly, we prove identifiablity of the system poles from these recursions which, in turn, allows us to prove identifiability of the noncausal AR parameters from certain output cumulants (see Section IV).

Lemma 1. Given any real or complex number ρ, it cannot be a zero of $H_2(z;m)$ for every m such that $q-p \le m \le q$ where $p=p_1+p_2$. •

Proof: is given in Appendix B. □

Remark 3.1. The key idea and the basic approach embodied in Lemma 1 was first suggested by Tugnait (see [2] and references therein). An "indirect" method was used by Tugnait to examine the zeros of $H_2(z;m)$. A more "direct" investigation of $H_2(z;m)$ in [10] has yielded sharper results. We follow [10] to prove Lemma 1 in Appendix B. □

Define the p+1-column vector

$$\bar{C}_f(\tau) := [\,C_3(\tau,q-p)\;\; C_3(\tau,q-p+1)\;\; \cdots\;\; C_3(\tau,q)\,]^T. \tag{3.1}$$

By (2.20) the following vector recursion also holds

$$\bar{C}_f(\tau) = -\sum_{j=1}^{p_1+\bar{p}_2} a_{cj}\,\bar{C}_f(\tau-j) \quad \text{for } \tau \ge L_f+1,\; L_f := 2q+0.5p_2(p_2-1). \tag{3.2}$$

Theorem 1. Form an infinite block Hankel matrix $\mathbf{H}_f$ with the elements {ij-th element: $\bar{C}_f(L_f-p_c+i+j-1)$} where $p_c := p_1+\bar{p}_2$. Then we have $p_1 \le \mathrm{rank}\{\mathbf{H}_f\} \le p_1+\bar{p}_2$, and $\bar{p}_f = \mathrm{rank}\{\mathbf{H}_f\}$ is the minimal order of the vector recursion (3.2). Moreover, if (3.2) is rewritten in its irreducible form as

$$\bar{C}_f(\tau) = -\sum_{j=1}^{\bar{p}_f} \bar{a}_{cj}\,\bar{C}_f(\tau-j), \tag{3.3}$$

then the polynomial

$$\bar{A}_c(z) := 1 + \sum_{j=1}^{\bar{p}_f} \bar{a}_{cj}\, z^{-j}. \tag{3.4}$$

always has p_1 of its roots at the causal poles α_{ci} $(1 \le i \le p_1)$ of (1). •

Proof: That rank$\{\mathbf{H}_f\}=p_f$ implies that the minimal order of the recursion (3.2) is p_f, and vice-versa, follows from Gantmacher[11,p.244] (see also [2]). Referring to (2.1) and (2.14), the irreducible order of the scalar recursion (2.20) equals the number of non-cancelled causal poles of the product $H(z)H_2(z^{-1};m)$ for a given m. The irreducible order of the vector recursion (3.2) equals p_c minus the number of cancelled causal poles of the product $H(z)H_2(z^{-1};m)$ *common* to all $m \in [q-p,q]$. By Lemma 1, the causal poles of H(z) do not cancel in the product $H(z)H_2(z^{-1};m)$ for every $m \in [q-p,q]$. This gives the lower bound of p_1 on $\bar{p}_f$. This also proves the assertion that p_1 roots of (3.4) are located at α_{ci} $(1 \le i \le p_1)$. [Note that there is no guarantee that the zeros of H(z) will not cancel the causal poles of $H_2(z^{-1};m)$.] The upper bound on $\bar{p}_f$ follows trivially: just in case none of the causal poles under discussion get cancelled. □

The above result (Theorem 1) implies that if we solve the linear equations (3.2) for an appropriate selection of the lags, we can recover all the causal poles α_{ci} $(1 \le i \le p_1)$ and some (or all) of the inverse anticausal pole products $(\alpha_{ack}\alpha_{acj})^{-1}$. Of course, we would not know *apriori* which poles are causal and which are inverse anticausal pole products; i.e., this *linear estimation* procedure is not practical. But the results do establish the identifiability of the causal poles from the output cumulants $\{\bar{C}_f(\tau),\ \bar{L}_f \le \tau \le \bar{L}_f+M,\ M \ge 2p_c,\ \bar{L}_f := L_f+1-p_c\}$. Moreover, nonlinear methods can be used for constrained parameter estimation.

A result similar to Theorem 1 relying on the recursion (2.21) establishes the identifiability of the p_2 anticausal poles of the model (1.1). Define the p+1-column vector

$$\bar{C}_b(\tau) := [\, C_3(-\tau,-q+p) \;\; C_3(-\tau,-q+p-1) \;\; \cdots \;\; C_3(-\tau,-q) \,]^T. \tag{3.5}$$

By (2.21) the following vector recursion also holds

$$\bar{C}_b(\tau) = -\sum_{j=1}^{p_2+\bar{p}_1} a_{acj}\, \bar{C}_b(\tau-j) \quad \text{for } \tau \ge L_b+1,\ L_b := 2q+0.5p_1(p_1-1). \tag{3.6}$$

Theorem 2. Form an infinite block Hankel matrix $\mathbf{H}_b$ with the elements {ij-th element: $\bar{C}_b(L_b-p_{ac}+i+j-1)$} where $p_{ac} := p_2+\bar{p}_1$. Then we have $p_2 \le \text{rank}\{\mathbf{H}_b\} \le p_2+\bar{p}_1$, and $\bar{p}_b=$ rank$\{\mathbf{H}_b\}$ is the minimal order of the vector recursion (3.6). Moreover, if (3.6) is rewritten in its irreducible form as

$$\bar{C}_b(\tau) = -\sum_{j=1}^{\bar{p}_b} \bar{a}_{acj}\, \bar{C}_b(\tau-j), \tag{3.7}$$

then the polynomial

$$\bar{A}_{ac}(z) := 1 + \sum_{j=1}^{\bar{p}_b} \bar{a}_{acj}\, z^{-j}. \tag{3.8}$$

always has p_2 of its roots at the anticausal poles α_{aci} $(1 \le i \le p_2)$ of (1). •

By using the above results we can establish the lower bounds to the maximum cumulant lags required to obtain consistent parameter estimates via cumulant matching; a similar result has been obtained for noncausal AR models in [3] where both second- and third- order cumulants are used.

IV. AR IDENTIFIABILITY

Let

$$\theta := (a_{11}, a_{12}, \cdots, a_{2p_1}, a_{21}, a_{22}, \cdots, a_{2p_2}, b_0, b_1, \cdots, b_q) \quad (\gamma_{3w} := 1.) \tag{4.1}$$

denote the p+q+1 vector of the unknowns, where we have set the driving noise third cumulant to be unity without any loss of generality. Further, let θ_0 denote the true value of θ that generates the given observations. We will use the notation $a_{ij}(\theta)$ to denote the parameter a_{ij} parametrized by θ and similarly for $C_3(\tau,m\,|\,\theta)$.

The main result is

Theorem 3. If $C_3(\tau,m\,|\,\theta)=C_3(\tau,m\,|\,\theta_0)$ for $q-p\le m\le q$ and $\bar{L}_f\le\tau\le\bar{L}_f+M_f$ where $M_f\ge 2p_c$ and $L_f:=L_f+1-p_c$, and if $C_3(-\tau,-m\,|\,\theta)=C_3(-\tau,-m\,|\,\theta_0)$ for $q-p\le m\le q$ and $L_b\le\tau\le L_b+M_b$ where $M_b\ge 2p_{ac}$ and $L_b:=L_b+1-p_{ac}$, then $a_{ij_k}(\theta)=a_{ij_k}(\theta_0)$ for k=1,2 and $1\le j_1\le p_1$, $1\le j_2\le p_2$.

Proof is by contradiction. If the claimed result is not true then we have a contradiction by either Theorem 1 or Theorem 2, because lack of the claimed equality of the AR coefficients would then imply lack of the recovery of the causal (resp. anticausal) system poles, thus contradicting Theorem 1 (resp. Theorem 2). ☐

V. CONCLUSIONS

Two Yule-Walker-type output cumulant recursions were derived for noncausal ARMA(p,q) signal models. These recursions were then used to derive upper and lower bounds on the ranks of the associated infinite block-Hankel matrices in terms of the system causal and anticausal poles. Finally, these results were used to prove the identifiabilty of the AR parameters from certain output cumulants.

APPENDIX A

Here we derive an expression for the product h(i)h(i+m) for m≥0 where h(i) is given by (2.4) and m is an integer. We define

$$h_{ma}(i) := [\sum_{j=0}^{q-p} c_j\,\delta(i-j)\,]\,u_{q-p} \tag{A1}$$

so that (2.4) can be rewritten as

$$h(i) = h_{ma}(i) + \bar{h}(i). \tag{A2}$$

We have

$$h_{ma}(i)h_{ma}(i+m) = \sum_{k=0}^{q-p+m} c_k\,c_{k+m}\,\delta(i-k) = 0 \quad \text{for } i \notin [0,q-p-m]. \tag{A3}$$

Also, by (tedious) algebraic manipulations, we have

$$h_{ma}(i)\bar{h}(i+m) = 0 \quad \text{for } i \notin [0, q-p], \tag{A4}$$

$$h_{ma}(i+m)\bar{h}(i) = 0 \quad \text{for } i \notin [-m, q-p-m]. \tag{A5}$$

Note that the right-sides of (A3)-(A5) vanish for all i if $q-p<0$.

Now we turn to the contribution of $\bar{h}(i)$. We have

$$\bar{h}(i)\bar{h}(i+m) = \sum_{t=1}^{L_c}\sum_{k=1}^{L_{ct}} r_{tk}(m)\, e_{ki}\, \beta_{ct}^{i}\, u_i + \sum_{t=1}^{L_{ac}}\sum_{k=1}^{L_{act}} \bar{r}_{tk}(m)\, e_{ki}\, \beta_{act}^{i}\, u_{-(i+m+1)} + \sum_{s=1}^{m} c_{1s}\, \delta(i+s) \tag{A6}$$

where (see also [2,Appendix])

$$\beta_{ct} := \alpha_{cj}\,\alpha_{cl} \quad \text{for } 1 \le j \le l \le n_c \text{ and } 1 \le t \le L_c \tag{A7}$$

$$\beta_{act} := \alpha_{acj}\,\alpha_{acl} \quad \text{for } 1 \le j \le l \le n_{ac} \text{ and } 1 \le t \le L_{ac} \tag{A8}$$

$$L_c := 0.5 n_c (n_c+1)\,; \quad L_{ct} := n_{cj} + n_{cl} - 1 \tag{A9}$$

$$L_{ac} := 0.5 n_{ac}(n_{ac}+1)\,; \quad L_{act} := n_{acj} + n_{acl} - 1 \tag{A10}$$

$$c_{1s} := \sum_{j=1}^{n_c}\sum_{k=1}^{n_{cj}}\sum_{l=1}^{n_{ac}}\sum_{r=1}^{n_{acl}} d_{jk}\, \bar{d}_{lr}\, e_{ki}\, e_{r(s+m)}\, \alpha_{acj}^{s}\, \alpha_{cl}^{s+m}\,. \tag{A11}$$

Notice that $c_{1s} \equiv 0$ whenever $p_i=0$ for either $i=1$ or $i=2$, i.e., whenever the ARMA model is purely causal or anticausal.

Combine (A3)-(A5) with (A11) to obtain the desired product $h(i)h(i+m)$ $(m \ge 0)$ whose Z-transform is given by (2.7) with c_{2i} appropriately defined.

APPENDIX B : Proof of Lemma 1

We will prove Lemma 1 only for the case when the system poles are distinct; however, our results continue to hold even if the poles are repeated (multiple). A proof for such a case can be devised following [10,Cor.1]. We have

$$H_2(z;m) = \frac{1}{2\pi(-1)^{1/2}} \oint u^{m-1} H(u)\, H(zu^{-1})\, du \tag{B1}$$

where

$$H(u) = [u^{p-q} \prod_{i=1}^{q} (u-\xi_i)][\prod_{i=1}^{p_1} (u-\alpha_{ci}) \prod_{j=1}^{p_2} (u-\alpha_{acj})]^{-1} \tag{B2}$$

$$H(zu^{-1}) = [z^{p-q}\, B \prod_{i=1}^{q} (u-z\,\xi_i^{-1})][A \prod_{i=1}^{p_1} (u-z\,\alpha_{ci}^{-1}) \prod_{j=1}^{p_2} (u-z\,\alpha_{acj}^{-1})]^{-1} \tag{B3}$$

$$B := \prod_{i=1}^{q} (-\xi_i)\ ; \quad A := [\prod_{i=1}^{p_1} (-\alpha_{ci})][\prod_{i=1}^{p_2} (-\alpha_{aci})]. \tag{B4}$$

The closed contour in (B1) is the unit circle running counter clockwise since H(z) is analytic in an annular region enclosing the unit circle. The poles of the integrand in (B1) encircled by the unit circle are located at

$$\alpha_{ci}\ (1 \le i \le p_1); \quad z\,\alpha_{aci}^{-1}\ (1 \le i \le p_2); \quad 0 \text{ if } m<q-p+1.$$

By the Cauchy Residue Theorem, $H_2(z;m)$ equals the sum of the residues at the enclosed poles. Therefore we have

$$H_2(z;m) = c_0\,\delta(m+p-q) + \sum_{i=1}^{p_1} c_{ci}(z)\,\alpha_{ci}^{m} + \sum_{i=1}^{p_2} c_{aci}(z)\,(z\,\alpha_{aci}^{-1})^{m} \quad \text{for } m\geq q-p, \tag{B5}$$

where

$$c_0 = B/A \tag{B6}$$

$$c_{ci} = (B/A)\,z^{p-q}\,\alpha_{ci}^{p-q+m-1} \prod_{j=1}^{q}(\alpha_{ci}-\xi_j)(\alpha_{ci}-z\,\xi_j^{-1})$$

$$\times\{\prod_{j=1,j\neq i}^{p_1}(\alpha_{ci}-\alpha_{cj})\prod_{j=1}^{p_2}(\alpha_{ci}-z\,\alpha_{acj}^{-1})\prod_{j=1}^{p_2}(\alpha_{ci}-\alpha_{acj})\prod_{j=1}^{p_1}(\alpha_{ci}-z\,\alpha_{cj}^{-1})\}^{-1} \quad (1\leq i\leq p_1) \tag{B7}$$

$$c_{aci} = (B/A)\,z^{p-q}\,(z\,\alpha_{aci}^{-1})^{p-q+m-1} \prod_{j=1}^{q}(z\,\alpha_{aci}^{-1}-\xi_j)(z\,\alpha_{aci}^{-1}-z\,\xi_j^{-1})$$

$$\times\{\prod_{j=1}^{p_1}(z\,\alpha_{aci}^{-1}-\alpha_{cj})\prod_{j=1,j\neq i}^{p_2}(z\,\alpha_{aci}^{-1}-z\,\alpha_{acj}^{-1})\prod_{j=1}^{p_2}(z\,\alpha_{aci}^{-1}-\alpha_{acj})\prod_{j=1}^{p_1}(z\,\alpha_{aci}^{-1}-z\,\alpha_{cj}^{-1})\}^{-1} \quad (1\leq i\leq p_2) \tag{B8}$$

Remark B1. If z is such that $z\,\alpha_{acj}^{-1}=\alpha_{ck}$ for some $j\in[1,p_2]$ and $k\in[1,p_1]$, then from (B7) and (B8) we have $c_{ck}\rightarrow\infty$ and $c_{acj}\rightarrow\infty$. This would seem to indicate that $z=\alpha_{ck}\,\alpha_{acj}$ is pole of $H_2(z;m)$ for all m. However, from (2.7) the poles of $H_2(z;m)$ are located at $\alpha_{ci}\,\alpha_{cn}$ $(i,n\in[1,p_1])$ and at $\alpha_{aci}\,\alpha_{acn}$ $(i,n\in[1,p_2])$ only. A closer examination of (B5)-(B8) reveals that for $z=\alpha_{acj}\,\alpha_{ck}$, we have $c_{ck}=-c_{acj}$ so that the corresponding terms vanish from (B5). □

In view of Remark B1, we modify (B5) as follows

$$H_2(z;m) = \begin{cases} c_0\,\delta(m+p-q) + \sum_{i=1}^{p_1} c_{ci}(z)\,\alpha_{ci}^{m} + \sum_{i=1}^{p_2} c_{aci}(z)\,(z\,\alpha_{aci}^{-1})^{m} \\ \qquad \text{for } z\neq\alpha_{acj}\alpha_{ck} \text{ for any } j\in[1,p_2],k\in[1,p_1] \\ c_0\,\delta(m+p-q) + \sum_{i=1,i\neq k}^{p_1} c_{ci}(z)\,\alpha_{ci}^{m} + \sum_{i=1,i\neq j}^{p_2} c_{aci}(z)\,(z\,\alpha_{aci}^{-1})^{m} \\ \qquad \text{for } z=\alpha_{acj}\alpha_{ck} \text{ for some } j\in[1,p_2],k\in[1,p_1] \end{cases} \tag{B9}$$

where $m\geq q-p$. We now turn to the proof of Lemma 1.

Proof of Lemma 1. Use (B9) and mimic the proof of [10, Lemma 1]. [In the above we tacitly assumed that there is no more than one α_{ac},α_c pair that yields the same product z. If it is not so, then the other pairs too vanish from $H_2(z;m)$ for that particular value of z leaving the conclusion of the Lemma unchanged.] •

REFERENCES

[1] A. Benveniste and M. Goursat, "Blind equalizers," *IEEE Trans. Communications,* vol. COM-32, pp. 871-883, Aug. 1984.

[2] J.K. Tugnait, "Recovering the poles from third-order cumulants of system output," *IEEE Trans. Automatic Control,* vol. AC-34, Oct. 1989.

[3] J.K. Tugnait, "Consistent parameter estimation for noncausal autoregressive models from higher-order statistics," *Automatica,* vol. 26, Jan. 1990.

[4] J.K. Tugnait, "Consistent order selection for noncausal autoregressive models from higher-order statistics," *Automatica,* vol. 26, to appear in early 1990 (accepted).

[5] K.S. Lii and M. Rosenblatt, "Deconvolution and estimation of transfer function phase and coefficients for nongaussian linear processes," *Annals of Statistics,* vol. 10, pp. 1195-1208, 1982.

[6] J.K. Tugnait, "Fitting noncausal autoregressive signal plus noise models to noisy non-Gaussian linear processes," *IEEE Trans. Automatic Control,* vol. 32, pp. 547-552, June 1987.

[7] C.L. Nikias and M.R. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *IEEE Proc.,* vol. 75, pp. 869-891, July 1987.

[8] M. Rosenblatt, *Stationary sequences and random fields.* Boston, MA: Birkhäuser, 1985.

[9] J.K. Tugnait, "Identification of linear stochastic systems via second- and fourth- order cumulant matching," *IEEE Trans. Information Theory,* vol. IT-33, pp. 393-407, May 1987.

[10] A. Swami and J.M. Mendel, "AR identifiability using cumulants," in *Proc. Workshop on Higher-Order Spectral Analysis,* Vail, CO, pp. 1-6, June 1989.

[11] F.R. Gantmacher, *Applications of the Theory of Matrices.* New York: Interscience, 1959.

SIGNAL DETECTION USING THE BISPECTRUM

P.A. NIELSEN
J.B. THOMAS

Department of Electrical Engineering
Princeton University
Princeton NJ 08544

ABSTRACT

In this paper, we show that the bi-covariance function is zero for a large number of distributions. Thus, if a process is symmetric, a zero bi-covariance function does not imply that the process is Gaussian. A distributed detector, based on the empirical bi-covariance function, is proposed. The asymptotic distribution and the asymptotic efficiency, as compared to a linear detector, are derived for each sensor output. Finally, sample bispectra are shown for both computer generated and real acoustic data.

1. INTRODUCTION

Recently there has been a growing interest in the bispectrum [1,3-5,8-10]. Since there are many reasons for this renewed interest, only a few of the most common ones are listed here. First, until recently, sufficient computational power has not been present to make computation and evaluation of higher order spectra, such as the bispectra, feasible. Second, higher order spectra are more difficult to interpret and understand than the power spectrum. Third, while the power spectrum is a powerful tool for investigating the linear mechanisms of a process, it is possible to investigate the nonlinear mechanisms by using a higher order spectra. And finally, it can be shown that only for Gaussian processes are all the higher order spectra zero. This may be quite useful in situations where one is interested in noise cancellation or in determining if a particular process is Gaussian.

There are many different applications for the bispectrum. In particular, Sadler et. al. [8,9] proposed an algorithm based on the bispectrum for image reconstruction. Subba Rao and Gabr [10] suggested a test based on the ratio of the bispectrum to a function of the power spectrum to determine if a process is linear. Hinich [5] used a similar test but claims it is capable of determining whether a linear process is Gaussian. Many more applications of the bispectrum, particularly to signal processing problems, can be found in [1].

The motivation for this work was to derive a test, based on the bispectrum, which could be used to discriminate between the simple hypotheses of the test defined in (1).

$$\begin{aligned} H_0 &: \mathbf{x} = \mathbf{n} \\ H_1 &: \mathbf{x} = \mathbf{n} + \theta \mathbf{s} \end{aligned} \tag{1}$$

Here $\mathbf{x}$ is an M sample realization of the random process $\mathbf{X}$ received by the detector, $\mathbf{n}$ is the realization of the noise random process $\mathbf{N}$, θ is a positive constant, and $\mathbf{s}$ contains the samples of the known, positive signal.

2. BACKGROUND AND DEFINITIONS

Traditionally, higher order spectra are defined in terms of the cumulants and not the moments of a process. One reason for defining the spectra in this manner is that the cumulants of order three and higher for a Gaussian process contain no information not provided in the first two cumulants, and thus are zero. Let n be the number real variables and let M define the joint characteristic function.

$$M(t_1,\ldots t_n) = E\{e^{-j(t_1X_1 + \cdots + t_nX_n)}\} \tag{2}$$

Let $r = k_1 + k_2 + \ldots + k_n$. Then the joint cumulants can be defined as

$$c_{k_1\ldots k_n} = (-j)^r \frac{\partial^r \ln M(t_1, t_2, \ldots t_n)}{\partial t_1{}^{k_1} \partial t_2{}^{k_2} \cdots \partial t_n{}^{k_n}} \Big|_{t_1=t_2=\cdots=t_n=0} \tag{3}$$

and the joint moments as

$$m_{k_1\ldots k_n} = (-j)^r \frac{\partial^r M(t_1, t_2, \ldots t_n)}{\partial t_1{}^{k_1} \partial t_2{}^{k_2} \cdots \partial t_n{}^{k_n}} \Big|_{t_1=t_2=\cdots=t_n=0} \tag{4}$$

It can be shown that, if the process is stationary and zero mean, then the second and third order cumulants and moments are identical. By definition, (4) is

$$m_{k_1\ldots k_n} \triangleq E\{X_1{}^{k_1} X_2{}^{k_2} \ldots X_n{}^{k_n}\} \tag{5}$$

Thus,

$$m_{10\cdots0} = E\{X_1\} = c_{10\cdots0} \tag{6}$$

and

$$m_{20\cdots0} = E\{X_1{}^2\} = c_{20\cdots0} \tag{7}$$

and

$$m_{110\cdots0} = E\{X_1X_2\} = c_{110\cdots0} \tag{8}$$

It is clear that (7) and (8) are the covariance function of **X**. For a stationary process, it is only necessary to retain the *time difference* between X_1 and X_2. Thus, by dropping the subscripts and adding a time shift, (8) can be rewritten as

$$m(\tau) \triangleq m_{110\cdots0} = E\{X(k)X(k+\tau)\} = c_{11\cdots0} \triangleq c(\tau) \tag{9}$$

Equivalently, we can write the third moments

$$m_{1110\cdots0} = E\{X_1X_2X_3\} = c_{1110\cdots0} \tag{10}$$

as

$$m(\tau_1, \tau_2) = E\{X(k)X(k+\tau_1)X(k+\tau_2)\} = c(\tau_1, \tau_2) \tag{11}$$

where τ_1 is the time shift between X_2 and X_1, and and τ_2 is the time shift between X_3 and X_1. Since dropping the subscripts in (9) and (11) will cause not any confusion, the more compact notation of (9) and (11) will be used. The power spectrum (12) and the bispectrum (13) are defined as the one-dimensional and the two-dimensional Fourier transforms of (9) and (11) respectively.

$$S(\omega) = \sum_{\tau=-\infty}^{\infty} c(\tau)e^{-j\omega\tau} \tag{12}$$

$$B(\omega_1,\omega_2) = \sum_{\tau_1=-\infty}^{\infty} \sum_{\tau_2=-\infty}^{\infty} c(\tau_1,\tau_2) e^{-j(\omega_1\tau_1 + \omega_2\tau_2)} \tag{13}$$

As with the power spectrum, the bispectrum has several symmetry conditions. For a real process, it can be shown that

$$c(n,m) = c(m,n) = c(-n,m-n) = c(n-m,-m) \tag{14}$$

Thus, if the two-dimensional plane is parsed into the six sectors shown in Figure 1, knowledge of the bi-covariance function in any one sector, including edges, is equivalent to knowledge of the complete bi-covariance function by using the symmetry relationships [1]. Examining (13), it is clear that

$$B(\omega_1,\omega_2) = B(\omega_2,\omega_1) = B(\omega_1,-\omega_1-\omega_2) = B(-\omega_1,-\omega_2)^* \tag{15}$$

and that the bispectrum, unlike the power spectrum, is usually complex. A difficult question with the bispectrum is its interpretation. Quite clearly, it represents the frequency decomposition of the third order moments or cumulants. A more illuminating interpretation is that of Nikias and Raghuveer [1]. They describe the power spectrum as representing "the contribution to the mean product of two Fourier components whose frequencies are the same," and describe the bispectrum as representing "the contribution to the mean product of three Fourier components where one frequency equals the sum of the other two."

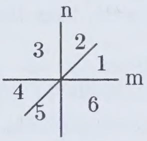


Figure 1 - Symmetry Regions for Third Order Moments [1]

3. RESULTS

3.1 Zero Bispectrum Cases

For a zero mean process, it can be shown that a normalized form of the bispectrum will be a constant if the process is linear, and will be zero if the process is Gaussian [5,10]. Many authors use these properties of the bispectrum to determine if a process is linear and Gaussian. However, the reverse implication of the second statement does not hold. That is, a zero bispectrum *does not* imply that a process is Gaussian.

Consider an independent, zero mean process $\mathbf{X}$ that is symmetric about the origin. It is clear that if the third moment is well defined and exists, then $E\{X_i^3\} = c(0,0) = 0$. Note in particular, if $\mathbf{X}$ is distributed as a Cauchy process, then this condition is not satisfied. In addition, terms of the form $c(n,0)$ and $c(n,m)$ will be zero since the multivariate density can be factored into a product of univariate densities, and by the symmetry constraint $E\{X_i\} = 0$. Thus, *any* process which is composed of independent samples and is symmetric about zero, and whose moments exist will have a zero bispectrum. Next, consider a nonlinear memoryless transformation of the process **X**. In addition, constrain the transformation to be odd symmetric about the origin and differentiable. As the transform simply pushes and pulls the axes symmetrically about zero, the bi-covariance function, and hence the bispectrum, of these processes will also be zero.

While the situation for independent processes is clear, what happens when the process is correlated is not obvious. Consider the generation of a process **Y** by a convolution of a white process **X** and a linear filter h. In the same manner that the power spectrum for the process **Y** can be derived, the bispectrum of **Y** can be shown to be [1]

$$B_Y(\omega_1,\omega_2) = H(\omega_1)H(\omega_2)H^*(\omega_1+\omega_2)B_X(\omega_1,\omega_2) \tag{16}$$

Hence, the bispectrum for **Y** will also be zero.

Finally, consider processes whose multivariate distributions satisfy Pearson's functional [7] and are expandable in canonical polynomials. From Brown [2], under these conditions the kth order conditional expectation can be written as a kth order polynomial. If the process **X** is symmetric about a zero mean, then terms of the form $c(0,0)$ will be zero. For one nonzero shift,

$$\begin{aligned} c(n,0) &= E\{X(t)^2X(t+n)\} = \int\int x_t^2 x_{t+n} f(x_t, x_{t+n})dx_t dx_{t+n} \\ &= \int x_t^2 f(x_t) \int x_{t+n} f(x_{t+n}\,|x_t)dx_{t+n}dx_t \end{aligned} \tag{17}$$

The inner integral will be a linear polynomial in x_t, e.g. $b_0 + b_1 x_t$. By symmetry constraints, polynomial will have no dc term; thus the inner integral will have the form $b_1 x_t$, where b_1 is a constant. Thus

$$c(n,0) = E\{X(t)^2X(t+n)\} = \int b_1 x_t^3 f(x_t) = 0 \tag{18}$$

Terms of the final form, $c(n,m)$, can be shown to be zero by using a similar technique.

3.2 Nonzero Bispectrum Case

Consider the situation when the symmetric process does not have a zero mean. Specifically let **X** be as represented as in H_1 and let the signal **s** be constant with strength s. In addition, let the noise process be zero mean. For $X(t) = N(t) + \theta s$, the third moment can be written as

$$\begin{aligned} m(n,m) &= E\{X(t)X(t+n)X(t+m)\} \\ &= c_{noise}(n,m) + \theta s\sigma^2\{\rho_n + \rho_{m-n} + \rho_m\} + (\theta s)^3 \end{aligned} \tag{19}$$

where $c_{noise}(n,m)$ is the noise bi-covariance, σ^2 is the noise variance, and $\rho_n = E\{N(t)N(t+n)\}/\sigma^2$. Note, that (19) is not a bi-covariance function since the mean of **X** is nonzero and thus the third moment is not equivalent to the third cumulant. However, we will abuse the notation and call $m(n,m) = c(n,m)$ as in (11). From Section 3.1, the noise bi-covariance function is zero. Thus (19) simplifies to

$$c(n,m) = \theta s\sigma^2\{\rho_n + \rho_{m-n} + \rho_m\} + (\theta s)^3 \tag{20}$$

If the noise has nonnegative correlation, for a real signal (20) will be nonzero with the same sign as the signal for all choices of n and m.

3.3 Detector Proposal and Analysis

To summarize, in Section 3.1 it was shown that the bispectrum will be zero for many different types of zero mean noise processes. In Section 3.2 it was shown that when a constant signal is added to the noise, if the noise has nonnegative correlation, then the bi-covariance function will be nonzero and will have the same sign as the signal. Thus, for a positive signal, the following test, at each sensor, is sufficient to discriminate between the two hypotheses in (1).

$$c(n,m) \begin{array}{c} \geq \\ < \end{array} T_{n,m} \quad \begin{array}{c} \rightarrow H_1 \\ \rightarrow H_0 \end{array} \tag{21}$$

Once all the local sensor decisions are made, the decisions must be fused together to form a global decision. Thus, (21) represents only the first step in the decision process. Specifically, each sensor would output a one if H_1 is accepted and would output a zero if H_0 is accepted. Note that the thresholds are dependent on the particular choice of n and m used. Thus, (21) represents a very complex system. Since our interest primarily has to do with time series analysis, consider the sample bi-covariance (22) and bispectrum (23) functions.

$$\hat{c}(n,m) = \sum_{i=1}^{M} x_i x_{i+n} x_{i+m} \tag{22}$$

$$\hat{B}(\omega_1,\omega_2) = X(\omega_1)X(\omega_2)X^*(\omega_1+\omega_2) \tag{23}$$

where $X(\omega)$ is the DFT of $\mathbf{x}$.

Before we proceed to derive the asymptotic distribution of the sample bi-covariance function as defined by (22), consider Figures 2 to 4 where the magnitude of some sample bispectra are plotted. If one has L different zero mean and properly bandlimited time series, say $\mathbf{x}^l$ for $l=1, 2, .., L$, then averaging together the L estimates of the bispectrum as defined by (23), will result in an unbiased, consistent, and minimum variance estimate of the bispectrum according to Kim and Powers [13]. Using (23), a single estimate of the bispectrum for a vector of data can be computed. For the figures, 20 different estimates of the bispectra were averaged. For Figure 2, computer generated dependent Gaussian noise with a variance of four was used. The correlation coefficients of the noise were $\rho_1 = 0.378$ and $\rho_2 = 0.27$ and the signal strength divided by the noise standard deviation was 0.5. The only significant peak in this figure occurs at the origin. When the signal is removed, the bispectrum is nearly zero.

In Figures 3 and 4, the bispectra are estimated using a segment of Arctic under-ice noise [12]. The particular segment used is highly correlated, has a variance of four, and has at least one narrowband component present. In addition, this data can be shown to be non-stationary. In Figure 3, the noise is added to a signal of strength 0.5, while in Figure 4 no signal is present. The same scale factor was used on each figure. Note, that the bispectrum works as a noise canceller.

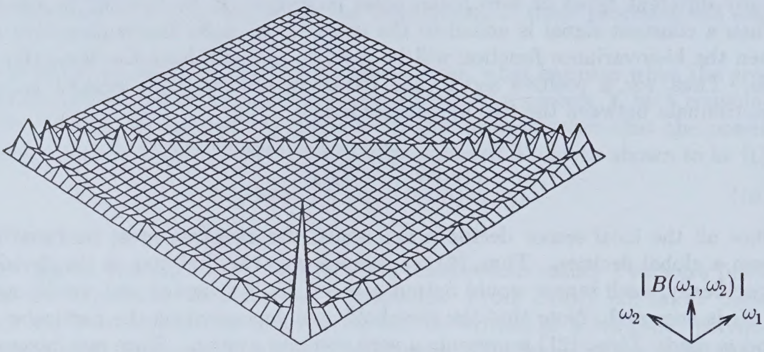


Figure 2 - Magnitude of Bispectrum n the First Quadrant for a Constant Signal in Dependent Gaussian Noise (spike amplitude $= 0.696$ at $\omega_1 = \omega_2 = 0$)

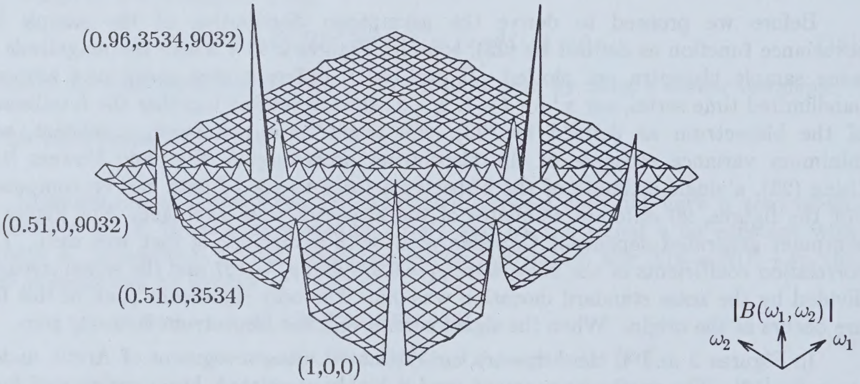


Figure 3 - Magnitude of Bispectrum in the First Quadrant for a Constant Signal in Records 4201-4203 Arctic Under-Ice Noise (amplitude of spike, ω_1, ω_2) $\omega_s = 12173.6$ rad/s.

Returning to the asymptotic analysis, three additional conditions on the process **X** are necessary to derive the asymptotic distribution of the sample bi-covariance function. First, $E|X_i|^{2+\delta}$ must be finite for some positive δ. Second, it is required that the variance change in the sense of Karamata [6], or

$$E\left\{\sum_{i=1}^{M} X_i\right\}^2 = Mh(M) \text{ and } \lim_{M\to\infty} \frac{h(tM)}{h(M)} = 1 \text{ for } \textit{any } t > 0. \tag{24}$$

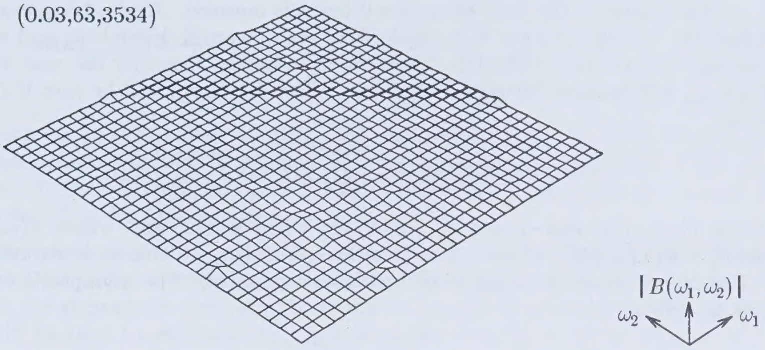


Figure 4 - Magnitude of Bispectrum in the First Quadrant for a Constant Signal in Records 4201-4203 of Arctic Under-Ice Noise (amplitude of spike, ω_1, ω_2)

Finally, it is required that samples which are separated by at least d time samples are independent. This conditions is known as d-dependence. In particular, we restrict the data to be d-dependent such that $d << M$. Under these assumptions, the Hoeffding-Robbins [11] theorem states that

$$\frac{1}{\sqrt{M}} \sum_{i=1}^{M} X_i \rightarrow N(0,\sigma^2) \tag{25}$$

We will analyze each of the different types of bi-covariance terms. Consider the case when $n = m = 0$. Let $y_i = x_i^3$. Then

$$\hat{c}(0,0) = \sum_{i=1}^{M} y_i \tag{26}$$

From Section 3.1 it is clear that $E\hat{c}(0,0) = 0$. Consider the variance of the above term.

$$\begin{aligned} V\hat{c}(0,0) = \sum_{i=1}^{M}\sum_{j=1}^{M} EY_iY_j &= \sum_{i=1}^{M} EY_i^2 + 2\sum_{i=1}^{M}\sum_{j=i+1}^{M} EY_iY_j \\ &= M\sigma_y^2 + 2M\sigma_y^2 \sum_{i=1}^{M} \rho_{y,i} - 2\sigma_y^2 \sum_{i=1}^{M} i\rho_{y,i} \end{aligned} \tag{27}$$

where $\rho_{y,i} = Cov(Y_0, Y_i)/{\sigma_y}^2$. Rewriting the above

$$V\hat{c}(0,0) = M\sigma_y^2 + 2M\sigma_y^2 \sum_{i=1}^{d-1} \rho_{y,i} - 2\sigma_y^2 \sum_{i=1}^{d-1} i\rho_{y,i} \tag{28}$$

Thus

$$h(M) = \sigma_y^2 + 2\sigma_y^2 \sum_{i=1}^{d-1} \rho_{y,i} - \frac{2\sigma_y^2}{M} \sum_{i=1}^{d-1} i\rho_{y,i} \tag{29}$$

Only the final term is dependent on M, and we see that the sum does vary in the sense of Karamata. Define $h_\infty = \lim_{M\to\infty} h(M)$. We can thus apply the Hoeffding-Robbins Theorem [11] and as a result $\hat{c}(0,0)/\sqrt{M}$ will asymptotically have a normal distribution with zero mean and variance h_∞.

Next, consider the case when $m = 0$ but n is nonzero. Again, let $y_i = x_i^2 x_{i+n}$. As before the original process **X** is stationary, zero mean, d-dependent, and satisfies the necessary conditions for Brown's Theorem [2] to apply. Consider the case when $n < d$. Then $\rho_{y,i}$ will be zero whenever $i \geq d + n$. When $n \geq d$, $\rho_{y,i}$ will be zero if $i \geq d$. Generally

$$V\hat{c}(n,0) = M\sigma_y^2 + 2M\sigma_y^2 \sum_{i=1}^{f(m,n,d)} \rho_{y,i} - 2\sigma_y^2 \sum_{i=1}^{f(m,n,d)} i\rho_{y,i} \tag{30}$$

where $f(m,n,d) = n{+}d{-}1$ and $\sigma_y^2 = E\{X_o^4 X_n^2\}$ when $n < d$; and where $f(m,n,d) = d{-}1$ and $\sigma_y^2 = E\{X^4\}E\{X^2\}$ when $n \geq d$. In either case, the new process as defined by **Y** has a dependence structure which is at most $n{+}d$-dependent. The asymptotic variance can now be defined as

$$h_\infty = \sigma_y^2 + 2\sigma_y^2 \sum_{i=1}^{f(m,n,d)} \rho_{y,i} \tag{31}$$

Applying the Hoeffding-Robbins Theorem [11], the asymptotic distribution of the normalized $\hat{c}(n,0)$ will be Gaussian with zero mean and variance defined by h_∞.

A similar type of analysis can be done for the final case when both n and m are nonzero and not equal. Finally, to get a feeling for the efficiency of the proposed detector the Asymptotic Relative Efficiency (ARE) between the the bi-covariance detector and the linear detector is derived. Define the efficacy of a detector l as

$$\xi_l = \lim_{M\to\infty} \frac{1}{M} \frac{\left[\frac{d}{d\theta} E\, l(y)\ |_{\theta=0}\right]^2}{Var(l(y))\ |_{\theta=0}} \tag{32}$$

For the bi-covariance detector with l defined by (26)

$$E\, l(y) = M\left((\theta s)^3 + 3\theta \bar{\rho}_x s \sigma_x^2\right) \tag{33}$$

where $\bar{\rho}_x = (\rho_{x,m} + \rho_{x,n} + \rho_{x,(n-m)})/3$, and σ_x^2 is the noise power of the received process **X**. Differentiating (33) and letting θ tend to zero results in

$$\left[\frac{d}{d\theta} E\, l(y)\ |_{\theta=0}\right]^2 = \left[M\, 3\bar{\rho}_x s \sigma_x^2\right]^2 \tag{34}$$

The variance of the bi-covariance detector can be written in the following form, where $f(m,n,d)$ is either $d{-}1$ or $n{+}d{-}1$ depending on the exact choices of n, m, and d being used. Thus

$$\xi_l = \lim_{M\to\infty} \frac{1}{M} \frac{\left[M\, 3\bar{\rho}_x s \sigma_x^2\right]^2}{M\,\sigma_y^2\left(1 + 2\sum_{i=1}^{f(m,n,d)} \rho_{y,i} - \frac{2}{M}\sum_{i=1}^{f(m,n,d)} i\,\rho_{y,i}\right)} = \frac{\left[3\bar{\rho}_x s \sigma_x^2\right]^2}{\sigma_y^2\left(1 + 2\sum_{i=1}^{f(m,n,d)} \rho_{y,i}\right)} \tag{35}$$

In an equivalent fashion the efficacy of the linear detector $l_{lin} = \sum_{i=1}^{M} x_i$ can be defined as

$$\xi_{lin} = \frac{s^2}{\sigma_x^2\left(1 + 2\sum_{i=1}^{d-1} \rho_{x,i}\right)} \tag{36}$$

Note, that the variance in (36) is computed using the knowledge that the data is correlated. The asymptotic relative efficiency of the bi-covariance detector to the linear detector is then

$$\mathrm{ARE}_{bi-cov,lin} = \frac{\xi_{bi-cov}}{\xi_{lin}} = \left[\,3\bar{\rho}_x \sigma_x^2\right]^2 \frac{\sigma_x^2(1+2\sum\limits_{i=1}^{d-1}\rho_{x,i}\,)}{\sigma_y^2(1+2\sum\limits_{i=1}^{f(m,n,d)}\rho_{y,i}\,)} \tag{37}$$

The above expression is relatively complex and requires knowledge of both the original process **X** and the new process **Y** which is a delayed and multiplied version of the original process. It is clear that the ARE will be dependent on the particular choices of n, m, and d and that the ARE can be zero.

From the previous discussion, we know that the $\hat{c}$'s will have a Gaussian distribution, asymptotically. For a fixed M, this is equivalent to restricting $n + d << M$ and M large (recall, the dependence between the terms in the sum is at most $n + d$). In general, (23) will be used to estimate the bispectrum and then inverted to estimate (22). Since the estimates for all n and m will be available simultaneously, consider as an alternative to (21) comparing the statistic,

$$\sum_n \sum_m \hat{c}(n,m)\, a_{n,m} \qquad M-1 \geq n \geq 0; \quad n \geq m \geq 0 \tag{38}$$

where the $a_{n,m}$ are real weights to some threshold T. Here, if (38) is greater than the threshold T hypothesis H_1 is accepted, else hypothesis H_0 is accepted. Note that the limits on the sum are defined by Sector 1 and the boundaries shown in Figure 1, with the restriction that $n + m \leq M-1$ due to the symmetry of the DFT. (Note that while (21) is a distributed detector, (38) is not. No local decisions at the sensors are being made). Thus, if the weights $a_{n,m}$ of (38) are chosen to be zero unless $n + d << M$, then the detector defined by (38) reduces to a weighted combination of correlated Gaussian random variables. The sum of correlated Gaussian random variables is Gaussian, thus the performance of (38) can be described by the classical result

$$\beta = 1 - \Phi\,(\Phi^{-1}(1-\alpha) - \gamma) \tag{39}$$

where α is the probability of false alarm, β is the power of the probability of detection, γ^2 is a generalized signal to noise ratio which relates the number of samples, the signal power, the noise power, and the correlation between the different terms in the sum defined in (26). Specifically, let $l(\mathbf{X})$ be the function defined by (38) with the weights chosen to be zero unless $n + d << M$. If E_1 denotes expectation taken with respect to hypothesis H_1, then

$$\gamma = \frac{E_1\{l\}}{[V\{l\}]^{1/2}} \tag{40}$$

4. CONCLUSIONS

It has been shown that the bi-covariance function is zero for a large number of symmetric distribution functions. This suggests that the bi-covariance function is a test of the symmetry of a process, and that it cannot be used to determine if a process is Gaussian. A detector, based on the empirical bi-covariance function, was proposed and expressions for the asymptotic distribution of the test and the asymptotic relative efficiency for this test as compared to the linear test for the sensor outputs are given. Due to the complexity of the expressions, no direct results are available for non-Gaussian environments. In addition, sample bispectra were given for both computer generated and real acoustic data.

5. ACKNOWLEDGEMENT

This work was supported by the Office of Naval Research under Contract N00014-88-K-0162 and by the National Science Foundation under Grant ECS87-13598.

6. REFERENCES

[1] C.L. Nikias and M.R. Raghuveer, "Bispectrum Estimation: A Digital Signal Processing Framework," *Proc. of the IEEE,* **75**(7), pp 869-891, 1987.

[2] J. Brown, "A Criterion for the Diagonal Expansion of a Second-Order Probability Distribution in Orthogonal Polynomials," *IRE Transactions on Information Theory,* Vol IT-4, pp 172, 1958.

[3] P.L. Brockett, M. Hinich, and G.R. Wilson," Bispectral Characterization of Ocean Acoustic Time Series: Nonlinearity and Non-Gaussianity," in *Topics in Non-Gaussian Signal Processing,* Eds E.J. Wegman, J.B. Thomas, and S.C. Schwartz, Springer-Verlag, NY, 1989.

[4] P.L. Brockett, M. Hinich, and G.R. Wilson," Non-Linear and Non-Gaussian Ocean Noise," *J. Acoust. Soc. Am.* **82** (4), pp 1386-1394, 1987.

[5] M.J. Hinich, "Testing for Gaussianity and Linearity of a Stationary Time Series," *J. Time Series Anal.* **3**(3), 169-176, 1982.

[6] M. J. Karamata, "Sur un Mode de Croissance Reguliere; Theorems Fondamentaux," *Bull. Soc. Math. de France,* Vol 61, pp. 55-62, 1933.

[7] H. Lancaster, *The Chi-Square Distribution,* Wiley, 1969.

[8] B.M. Sadler and G.B. Giannakis, "Image Sequence Analysis and Reconstruction from the Bispectrum," *Proc. 1989 Conf. on Inf. Sci. and Syst.,* Baltimore, MD., March 1989.

[9] B.M. Sadler, "Shift and Rotation Invariant Object Reconstruction using the Bispectrum," *Proc. ASSP Workshop on Higher-Order Spectral Analysis,* June 1989.

[10] T. Subba Rao and M. Gabr, "A Test for Linearity of Stationary Time Series," *J. Time Series Anal.* **1**, 145-158, 1980.

[11] A. Traganitis, *Narrow-Band Filtering, Estimation, and Detection of Non-Gaussian Processes,* Ph.D. Thesis Princeton University 1974.

[12] R. Dwyer,"Arctic Ambient Noise Statistical Measurement Results and their Implications to Sonar Performance Improvements," in *Proceedings of Conference on Undersea Ambient Noise* (SACLANT ASW Research Centre, La Spezia, Italy), 11-14 May 1982.

[13] Y.C. Kim and E.J. Powers, "Digital Bispectral Analysis and Its Applications to Nonlinear Wave Interactions," *Proc. IEEE Plasma Science,* 120-131, 1979.

ON AN ASPECT OF OPTIMAL NONLINEAR ESTIMATION

ERIC B. HALL
Department of Electrical Engineering
Southern Methodist University
Dallas, Texas 75275

GARY L. WISE
Department of Statistics
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

This paper considers the problem of optimal estimation of a random variable X based on an observation denoted by a random vector Y. A mild restriction on the regular conditional distribution function of X given $\sigma(Y)$ is presented which ensures that $E[\Phi(X-g(Y))]$ is minimized for any cost function Φ that is nonnegative, even, and convex. Further, we show that given *any* real valued Borel measurable function there exist random variables X and Y, possessing a joint density function, so that the chosen function is the optimal estimator, with respect to any nonnegative, even, convex cost function Φ, of the random variable X as a function of the random variable Y. Finally, the estimation of X via a random variable measurable with respect to a given σ–subalgebra is treated.

INTRODUCTION

In this paper we consider the problem of estimation with respect to nontraditional cost functions. In an estimation problem one is often confronted with two concerns in choosing a cost function: the concern that the cost function adequately reflects the cost one wishes to associate with an error, and the concern that the cost function results in a problem which one finds to be mathematically tractable. Traditional cost functions, such as the extremely popular mean square error cost function, are usually chosen solely on the basis of the second of the above two concerns. As a result, the fidelity demands of the specific problem under consideration are rarely relied upon, and, in fact, are often not even considered, when determining the cost function which will be used. This sacrifice of suitability for mathematical ease in the choice of a cost function should be the cause of some concern since the traditional choices are unsuitable for many problems in estimation.

This paper lessens this problem by extending the domain of mathematical tractability to include many cost functions which, even though pertinent to the subjective demands of many problems, have in the past been excluded from consideration.

DEVELOPMENT

In 1958, Seymour Sherman published a paper entitled "Non–Mean–Square Error Criteria" [1] in which he proposed conditions on a conditional distribution which would allow for the simultaneous minimization of a large family of cost functions. Sherman's effort, although ambitious and widely quoted, lacked the requisite attention to mathematical details and ignored several necessary measure theoretic subtleties. For example, Sherman's use of the first lemma in [1] results in two integrals which are undefined without the additional assumption that the conditional distribution function under consideration is regular, and, further, [1] incorrectly states that this same lemma holds for any distribution function that is absolutely continuous.

Several subsequent efforts to provide a correct development of the proposition in [1] also proved inadequate. For example, Viterbi [2, pp.308–310], Sakrison [3, pp.10–12], and Van Trees [4, p.61] attempted proofs using integration by parts even though the conditions placed on the cost function by the aforementioned authors were insufficient to allow such a method to be used. After establishing some notation and reviewing some definitions, we will present an example which illustrates this difficulty.

Throughout this paper, almost everywhere (a.e.) will be with respect to Lebesgue measure unless noted otherwise, for a topological space T, we will let $\mathcal{B}(T)$ denote the Borel subsets of T, and $I_S(\cdot)$ will denote the indicator function of the set S. Also, recall that a probability distribution function $F:\mathbf{R}\to[0, 1]$ is said to be unimodal about $\gamma \in \mathbf{R}$ if F is convex on $(-\infty, \gamma)$ and concave on (γ, ∞), and a probability distribution function $F:\mathbf{R} \to [0, 1]$ is said to be symmetric if for all real x, $F(x) = 1 - \lim_{h\downarrow 0} F(-x-h)$. If k is a positive integer and $Y_1, \ldots, Y_k$ are k random variables defined on a common probability space, the random vector $Y = [Y_1, \ldots, Y_k]^T$ induces a probability measure on $\mathcal{B}(\mathbf{R}^k)$; we will denote this resulting measure by the notation μ_Y. Finally, we recall that for a random variable X and a σ–subalgebra $\mathcal{A}$, a regular conditional distribution function $F:\mathbf{R} \times \Omega \to [0, 1]$ always exists [5, pp.263-264]; such a function is characterized by the following two conditions: for each $\omega \in \Omega$, $F(\cdot, \omega)$ is a probability distribution function, and, for each $x \in \mathbf{R}$, $F(x, \omega) = P(X \leq x \mid \mathcal{A})(\omega)$ a.s.

Consider two random variables X and Y defined on some probability space $(\Omega, \mathcal{S}, P)$ so that X and Y are mutually Gaussian random variables with zero means, unit variances, and correlation coefficient ρ. Recall that $E[X|Y=y] = \rho y$ a.e. Let $f:[0,1] \to [0,1]$ be the familiar Cantor–Lebesgue function defined in [6, p.35], and let a cost function $\Phi:\mathbf{R} \rightarrow [0,\infty)$ be defined as follows:

$$\Phi(x) = \begin{cases} \Phi(-x) \text{ if } x < 0 \\ (n-1) + f(x-n+1) \text{ if } x \in [n-1,n);\ n \in \mathbf{N}. \end{cases}$$

Clearly, $\Phi(\cdot)$ is even. Further, the well known properties of f (given in [6]) immediately imply that the cost function Φ is also nonnegative, nondecreasing on $[0, \infty)$, continuous, and singular (i.e. its derivative vanishes a.e.) The usual attempt to prove Theorem 1 using integration by parts is based upon a claim that

$$E[\Phi(X-\alpha)|Y=y] - E[\Phi(X-\rho y)|Y=y] = \lim_{x\to\infty} \Phi(z)u(z)\Big|_0^x - \int_0^\infty \frac{d\Phi(t)}{dt}\, u(t)\, dt$$

where u(t) is defined by

$$u(t) = \int_0^{\alpha-\rho y} [\, g(s + t + \rho y \mid Y=y) - g(s - t + \rho y \mid Y=y) \,]\, ds,$$

where $g(x|Y=y)$ is a conditional Gaussian density function of X given Y = y, where y is some fixed real number, and where $\alpha > \rho y$. The first term above,

$$\lim_{x\to\infty} \Phi(z)u(z)\Big|_0^x = \lim_{x\to\infty} \Phi(x)u(x) - \Phi(0)u(0),$$

is equal to zero in this case and equals zero in the indicated references due to restrictions placed on the conditional density. The problem which now arises is that the second term,

$$\int_0^\infty \frac{d\Phi(t)}{dt}\, u(t)\, dt,$$

is also equal to zero since $\frac{d\Phi(t)}{dt} = 0$ a.e. Thus, if the above claim were true then it must follow that, for this example, we have $E[\Phi(X-\alpha)|Y=y] = E[\Phi(X-\rho y)|Y=y]$, or that $E[\Phi(X-\alpha)-\Phi(X-\rho y)|Y=y] = 0$ for any choice of $\alpha > \rho y$. This last equation, which for almost all α is false, yields a contradiction to the claim that was made earlier. Furthermore, the continuity of the cost function and the conditional density implies that no specious appeal to atomic measures will salvage a method based on integration by parts. Thus, we see that, due to the great generality yielded by Sherman's conditions on the allowable cost functions, an attempt to prove Theorem 1 via a method which utilizes the previous claim is futile, and hence the developments of Sherman's proposition given in [2, pp.308–310], [3, pp.10–12], and [4, p.61] are incorrect.

In [7] we put Sherman's result on a firm mathematical basis and explored several extensions and practical consequences of his proposal. Theorem 1, stated below and proved in [7], provides a correct statement of Sherman's original proposal. Here then is the correct version of "Sherman's theorem."

Theorem 1: Let $k \in \mathbf{N}$, $(\Omega, \mathcal{S}, P)$ be a probability space, and $X, Y_1, \ldots, Y_k$ be random variables defined on $(\Omega, \mathcal{S}, P)$, with X integrable. Let $M:\mathbf{R}^k \to \mathbf{R}$ be a Borel measurable function such that $M[Y_1(\omega), \ldots, Y_k(\omega)] = E[X \mid Y_1, \ldots, Y_k](\omega)$ a.s., and assume that there exists a regular conditional distribution function of X conditioned on $\sigma(Y_1, \ldots, Y_k)$, $F:\mathbf{R} \times \Omega \to [0,1]$, such that $F(x+M[Y_1(\omega), \ldots, Y_k(\omega)],\omega)$, as a function of x with ω fixed, is unimodal about the origin and symmetric. Then $M[Y_1, \ldots, Y_k]$ minimizes the quantity $E[\Phi(X-f(Y_1, \ldots, Y_k))]$ over all Borel measurable functions $f:\mathbf{R}^k \to \mathbf{R}$ where $\Phi:\mathbf{R} \to [0, \infty)$ is even and nondecreasing on $[0, \infty)$.

Thus, Sherman's result as correctly stated in Theorem 1 requires a regular conditional distribution function that, when properly shifted, is symmetric and unimodal about the origin and a cost function that is nonnegative, even, and nondecreasing to the right of the origin. It is easy to see that if in Theorem 1 we let $k=1$ and X and Y be mutually Gaussian random variables then the resulting regular conditional distribution function is symmetric and unimodal about a version of $E[X|Y](\omega)$ for any fixed ω. This special case explains why Sherman's result is often invoked to add a token claim of generality to papers that only consider Gaussian distributions. When one attempts to venture outside this somewhat limited arena, however, the conditions which Theorem 1 places on the regular conditional distribution function immediately begin to feel overly restrictive. After all, how comfortable should we be with the assumption that the regular conditional distribution function under consideration is unimodal about the conditional mean? The conditions on the cost function, on the other hand, are extremely nonrestrictive and, in fact, allow for many interesting, albeit impractical, choices. For example, the cost function given by

$$\Phi(x) = \int_0^{|x|} I_C(t)\, dt,$$

where C denotes a Cantor subset of $[0, \infty)$ of positive Lebesgue measure, satisfies the conditions of Theorem 1. This imbalance suggests the possibility of lessening the restrictions on the regular conditional distribution function by perhaps slightly increasing the restrictions imposed on the cost function. The following lemma will allow us to present such a result.

Lemma 2: Assume that F is a symmetric probability distribution function and that $\Phi:\mathbf{R} \to [0,\infty)$ is even and convex. Then $\int_{\mathbf{R}} \Phi(x)\, dF(x) \leq \int_{\mathbf{R}} \Phi(x-\alpha)\, dF(x)$ for all $\alpha \in \mathbf{R}$.

Proof: Since Φ is convex, $\Phi(x) \leq \frac{1}{2}\Phi(x-\alpha) + \frac{1}{2}\Phi(x+\alpha)$. Further, since F is symmetric and Φ is even, $\int_{\mathbf{R}} \Phi(x+\alpha)\, dF(x) = \int_{\mathbf{R}} \Phi(x-\alpha)\, dF(x)$. Thus, it follows that

$$\int_{\mathbf{R}} \Phi(x)\, dF(x) \leq \frac{1}{2}\int_{\mathbf{R}} \Phi(x-\alpha)\, dF(x) + \frac{1}{2}\int_{\mathbf{R}} \Phi(x+\alpha)\, dF(x) = \int_{\mathbf{R}} \Phi(x-\alpha)\, dF(x).$$

Q.E.D.

As we will now show, Lemma 2 allows a result similar to that of Theorem 1 to be stated for a much less restrictive family of regular conditional distribution functions by slightly restricting the family of allowable cost functions. In particular, we will be able to drop the restriction that the conditional distribution function be unimodal by requiring that the cost function, in addition to the previous restrictions, also be convex. Notice that requiring the cost function to be even and convex implies that it is also nondecreasing to the right of the origin. Also, notice that a close examination of the proof of Lemma 2 reveals that convexity appears to be a slightly more restrictive condition than is needed. In fact, all that is required is that the cost function Φ be midpoint convex, i.e. that $\Phi((x+y)/2) \leq (\Phi(x)+\Phi(y))/2$ for all $x,y \in \mathbf{R}$. Although, in general, midpoint convexity does not imply convexity, midpoint convexity and monotonicity do imply convexity. Thus, we do not decrease the family of allowable cost functions by stipulating convexity in place of midpoint convexity.

The following theorem provides the same conclusion as Theorem 1 while greatly relaxing the conditions which Theorem 1 placed on the regular conditional distribution function. Although the family of allowable cost functions is decreased from those considered in Theorem 1, the family under consideration below is still quite large and diverse.

Theorem 3: Let $k \in \mathbf{N}$, $(\Omega, \mathcal{S}, P)$ be a probability space, and $X, Y_1, \ldots, Y_k$ be random variables defined on $(\Omega, \mathcal{S}, P)$, with X integrable. Let $M:\mathbf{R}^k \to \mathbf{R}$ be a Borel measurable function such that $M[Y_1(\omega), \ldots, Y_k(\omega)] = E[X \mid Y_1, \ldots, Y_k](\omega)$ a.s., and assume that there exists a regular conditional distribution function of X conditioned on $\sigma(Y_1, \ldots, Y_k)$, $F:\mathbf{R} \times \Omega \to [0,1]$, such that $F(x+M[Y_1(\omega), \ldots, Y_k(\omega)],\omega)$, as a function of x with ω

fixed, is symmetric. Then $M[Y_1, \ldots, Y_k]$ minimizes the quantity $E[\Phi(X - f(Y_1, \ldots, Y_k))]$ over all Borel measurable functions $f:\mathbf{R}^k \to \mathbf{R}$ where $\Phi:\mathbf{R} \to [0, \infty)$ is even and convex.

Proof: Lemma 2 implies that for each fixed ω and for $\alpha \in \mathbf{R}$,

$$\int_{\mathbf{R}} \Phi(x)\, dF(x+M(Y_1(\omega), \ldots, Y_k(\omega)),\omega) \leq \int_{\mathbf{R}} \Phi(x-\alpha)\, dF(x+M(Y_1(\omega), \ldots, Y_k(\omega)),\omega).$$

Changing variables yields

$$\int_{\mathbf{R}} \Phi(x-M(Y_1(\omega), \ldots, Y_k(\omega)))\, dF(x,\omega) \leq \int_{\mathbf{R}} \Phi(x-\alpha-M(Y_1(\omega), \ldots, Y_k(\omega)))\, dF(x,\omega).$$

Let $g:\mathbf{R}^k \to \mathbf{R}$ be a Borel measurable function by which X is to be estimated and $E[\Phi(X-g(Y_1, \ldots, Y_k))]$ minimized. Note that $E[\Phi(X-g(Y_1, \ldots, Y_k))] = E[\, E[\, \Phi(X-g(Y_1, \ldots, Y_k)) \mid \sigma(Y_1, \ldots, Y_k)\,]\,]$. From the preceding inequality and [8, p.79], the inner expectation, and thus the above expression, is minimized when $g(Y_1, \ldots, Y_k) = M(Y_1, \ldots, Y_k)$.

Q.E.D.

We will next present a useful corollary to Theorem 3. Let $k \in \mathbf{N}$, $(\Omega, \mathcal{S}, P)$ be a probability space, X be a random variable defined on $(\Omega, \mathcal{S}, P)$, and Y be a random vector defined on $(\Omega, \mathcal{S}, P)$ taking values in $\mathbf{R}^k$. Recall that $F(x|Y=y)$ is said to be a regular conditional distribution function for X given Y=y if for each fixed $y \in \mathbf{R}$, $F(x|Y=y)$ is a probability distribution function as a function of x, and for each fixed $x \in \mathbf{R}$, $F(x|Y=y)$ is a version of the regression function $E[I_{(-\infty,\, x]}(X)|Y=y]$. Further, recall that a regular conditional distribution function for X given Y=y always exists [*cf.* 9, pp.372–376]. The next corollary, which follows straightforwardly from Theorem 3, removes the need to work on the underlying probability space.

Corollary 4: Let $k \in \mathbf{N}$, $(\Omega, \mathcal{S}, P)$ be a probability space, X be an integrable random variable defined on $(\Omega, \mathcal{S}, P)$, Y be a random vector defined on $(\Omega, \mathcal{S}, P)$ taking values in $\mathbf{R}^k$, and $M:\mathbf{R}^k \to \mathbf{R}$ be any Borel measurable function equal a.e.$[\mu_Y]$ to $E[X|Y=y]$. Further, assume that, as a function of x with y fixed, a regular conditional distribution function of X given Y=y, denoted by $F(x|Y=y)$, is such that $F(x+M[y]\, |Y=y)$ is symmetric. Then $g(y) = M[y]$ minimizes $E[\Phi(X-g(Y))]$ over all Borel measurable functions $g:\mathbf{R}^k \to \mathbf{R}$ where $\Phi:\mathbf{R} \to [0,\infty)$ is even and convex.

Theorem 3 and Corollary 4 thus offer the trade off that was suggested earlier. In particular, by slightly increasing the restrictions on the family of allowable cost functions, we may drop the requirement that the regular conditional distribution function be unimodal, which seems overly restrictive from a utilitarian viewpoint. Further, the family of cost functions considered by Theorem 3 and Corollary 4, although smaller than the family which Sherman considered, is still quite large and, in fact, includes the L^p norm for $1 \leq p < \infty$.

The following example illustrates the usefulness of Theorem 3 and Corollary 4 and shows how these results may be applied to non–Gaussian distributions. In particular, given *any* real valued Borel measurable function $g(\cdot)$, we show that there exists a random variable X and a random variable Y, possessing a joint density function, so that $E[\Phi(X-h(Y))]$ is minimized when $h(Y) = g(Y)$ for any cost function Φ that is nonnegative, even, and convex.

Example: Let $g:\mathbf{R} \to \mathbf{R}$ be Borel measurable and define

$$f(x,y) = \frac{1}{8}\exp(-\exp(|y|)\,|x - g(y) + K|) + \frac{1}{8}\exp(-\exp(|y|)\,|x - g(y) - K|)$$

where K is some fixed real number. Note that f(x,y) is a joint probability density function since $\int_{\mathbf{R}}\int_{\mathbf{R}} f(x,y)\,dx\,dy$

$$= \int_{\mathbf{R}}\int_{\mathbf{R}} \frac{1}{8}\exp(-\exp(|y|)\,|x - g(y) + K|) + \frac{1}{8}\exp(-\exp(|y|)\,|x - g(y) - K|)\,dx\,dy$$

$$= \int_{\mathbf{R}}\int_{\mathbf{R}} \frac{1}{4}\exp(-\exp(|y|)\,|z|)\,dz\,dy = \int_{\mathbf{R}} \frac{1}{2}\exp(-|y|)\,dy = 1.$$

Let X and Y be random variables such that the pair (X, Y) has a joint density function given by f(x, y). Notice from the above calculation that the second marginal density of f(x, y) is given by $f_Y(y) = \frac{1}{2}\exp(-|y|)$. Also, notice that $f(x + g(y), y)$, as a function of x with y fixed, is even. Recall that a version of $E[X|Y=y]$ is given by

$$\int_{\mathbf{R}} x\,\frac{f(x,y)}{f_Y(y)}\,dx.$$

Fixing this version and using the above expression for $f_Y(y)$ implies that $E[X|Y=y]$

$$= 2\exp(|y|)\int_{\mathbf{R}} \frac{x}{8}\left(\exp(-\exp(|y|)\,|x - g(y) + K|) + \exp(-\exp(|y|)\,|x - g(y) - K|)\right)dx$$

$$= 2\exp(|y|)\int_{\mathbf{R}} ((z + g(y) - K) + (z + g(y) + K))\,\frac{1}{8}\exp(-\exp(|y|)\,|z|)\,dz$$

$$= 2\exp(|y|)\,(2(g(y)) + K - K)\,\frac{1}{4\exp(|y|)} = g(y).$$

Since f(x, y), as a function of x for y fixed, is even about g(y), it follows that the conditional density function $\frac{f(x,y)}{f_Y(y)}$ shares this same property. Thus, it is easy to see that the associated regular conditional distribution function, when properly shifted, is symmetric (and not unimodal if K is nonzero).

Corollary 4 may thus be applied to see that h(y) = g(y), which we recall was an arbitrary Borel measurable function, minimizes $E[\Phi(X-h(Y))]$ over all Borel measurable functions $h:\mathbf{R} \to \mathbf{R}$ where $\Phi:\mathbf{R} \to [0,\infty)$ is even and convex. Notice that this example illustrates the applicability of Theorem 3 in a situation where Theorem 1 would not apply, and, in addition, demonstrates the applicability of these results to non-Gaussian distributions. Notice further that this example also points out that the existence of a joint density function in no way guarantees that a regression function will obey any regularity property other than Borel measurability.

THE GENERAL CASE

The preceding development was concerned with an attempt to estimate the integrable random variable X based on a Borel measurable function of the random variables $Y_1, \ldots, Y_k$, where k is a positive integer. In this case, our estimate was a $\sigma(Y_1, \ldots, Y_k)$–measurable random variable. In many cases, we might wish to estimate X by a random variable which is measurable with respect to some other σ–algebra. For example, consider the case where $\{Y_t: t \in [0,1]\}$ is a random process and we wish to estimate X via a random variable which is $\sigma(\{Y_t: t \in [0,1]\})$–measurable. Also, consider the case where H is a real, separable Hilbert space and Z is an H–valued random variable; here we might wish to estimate X via a σ(Z)–measurable random variable. In the general case, Z could be a random object; that is, a random variable taking values in a measurable space $(G, \mathcal{G})$, and we would be interested in estimating X via a random variable which is measurable with respect to $\sigma(Z) = Z^{-1}(\mathcal{G})$.

The following theorem addresses the estimation of X via a random variable which is measurable with respect to a σ-subalgebra of $\mathcal{S}$.

Theorem 5: Let $(\Omega, \mathcal{S}, P)$ be a probability space, $\mathcal{A}$ be a σ–subalgebra of $\mathcal{S}$, and X be a random variable defined on $(\Omega, \mathcal{S}, P)$ such that X is integrable. For each $\omega \in \Omega$, let $M(\omega) = E[X|\mathcal{A}](\omega)$, and assume that there exists a regular conditional distribution function of X conditioned on $\mathcal{A}$, $F:\mathbf{R} \times \Omega \to [0,1]$, such that $F(x+M(\omega),\omega)$, as a function of x with ω fixed, is symmetric. Then M minimizes the quantity $E[\Phi(X-\widehat{X})]$ over all $\mathcal{A}$–measurable random variables $\widehat{X}$, where $\Phi:\mathbf{R} \to [0, \infty)$ is even and convex.

Proof: Lemma 2 implies that for each fixed ω and for $\alpha \in \mathbf{R}$,

$$\int_{\mathbf{R}} \Phi(x)\, dF(x+M(\omega),\omega) \le \int_{\mathbf{R}} \Phi(x-\alpha)\, dF(x+M(\omega),\omega).$$

Changing variables yields

$$\int_{\mathbf{R}} \Phi(x-M(\omega))\, dF(x,\omega) \le \int_{\mathbf{R}} \Phi(x-\alpha-M(\omega))\, dF(x,\omega).$$

Let $\widehat{X}$ be an $\mathcal{A}$–measurable random variable by which X is to be estimated and $E[\Phi(X-\widehat{X})]$ minimized. Note that $E[\Phi(X-\widehat{X})] = E[\, E[\, \Phi(X-\widehat{X}) \mid \mathcal{A}\,]\,]$. From the preceding inequality and [8, p.79], the inner expectation, and thus the above expression, is minimized when $\widehat{X} = M$.

Q.E.D.

CONCLUSION

Thus, in conclusion, this paper has considered the problem of optimal estimation of a random variable X based on an observation denoted by a random vector Y. We gave a mild restriction on the regular conditional distribution function of X given $\sigma(Y)$ which ensured that $E[\Phi(X-g(Y))]$ is minimized for any cost function Φ that is nonnegative, even, and convex. Further, we showed that given *any* real valued Borel measurable function there exist random variables X and Y, possessing a joint density function, so that the chosen function is the optimal estimator, with respect to any of the cost functions described above, of the random variable X based on the random variable Y. The results were then extended to estimation of X based upon a random variable which was measurable with respect to a given σ–subalgebra.

Finally, we hope that the results presented in this paper will contribute to a concept of estimation theory based on more meaningful cost functions than the overused and often abused mean square cost function. Despite the popularity of mean square estimation, its use is often a signal that the fidelity considerations of the problem have been ignored which, of course, should cast some doubt on the quality of the estimate. For those who still insist upon using mean square estimation techniques we note in closing that, even for bounded random variables, conditional expectation fails to minimize mean square error [10].

ACKNOWLEDGEMENT

This research was supported by the Air Force Office of Scientific Research under Grant AFOSR-86-0026.

REFERENCES

[1] S. Sherman, "Non–Mean–Square Error Criteria," *IRE Trans. Inform. Theory,* vol. IT–4, pp. 125–126, Sept. 1958.

[2] A. J. Viterbi, *Principles of Coherent Communication.* New York: McGraw–Hill, 1966.

[3] D. J. Sakrison, *Notes on Analog Communication.* New York: Van Nostrand Reinhold Company, 1970.

[4] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part 1.* New York: Wiley, 1968.

[5] Ash, R., *Real Analysis and Probability.* New York: Academic Press, 1972.

[6] R.L. Wheeden and A. Zygmund, *Measure and Integral.* New York: Dekker, 1977.

[7] E. B. Hall and G. L. Wise, "Simultaneous Optimal Estimation Over a Family of Fidelity Criteria," in *Proceedings of the 1987 Conference on Information Sciences and Systems,* Baltimore, MD, March 25–27, 1987, pp. 65–70.

[8] Breiman, L., *Probability.* Reading, Massachusetts: Addison-Wesley, 1968.

[9] Laha, R. G. and V. K. Rohatgi, *Probability Theory.* New York: John Wiley, 1979.

[10] G. L. Wise, "A Note on a Common Misconception in Estimation," *Systems and Control Letters,* vol. 5, pp. 355–356, April 1985.

SEQUENTIAL COMPUTATION BASED ON PIPELINED ACCESS TO MEMORY[1]

F. LUCCIO* and L. PAGLI **
* Dipartimento di Informatica, Università di Pisa, 56100 Pisa, Italy
** Dipartimento di Informatica e Applicazioni, Università di Salerno, 84081 Baronissi, Italy (Present address: National University of Somalia, Mogadishu, Somalia)

Abstract. We introduce a new sequential model of computation called Logarithmic Pipelined Model (LPM), in which a RAM processor of fixed size has pipelined access to a memory of m cells in time logm. We are motivated by the study of asymptotic complexity, since the memory cannot be accessed in constant time as m goes to infinity. For a problem Π of size n, $\Pi \in \mathbf{P}$, we denote by S(n) the time required by the best sequential algorithm, and by T(n) be the time required by an algorithm in LPM. Letting O(logn)=O(logm), we define the following classes: $LP_0 = \{\Pi \in \mathbf{P}: T(n) \in O(S(n))\}$, $LP_1 = \{\Pi \in \mathbf{P}: T(n) \in O(S(n) \log\log n)\}$, $LP_2 = \{\Pi \in \mathbf{P}: T(n) \in O(S(n)\log n/\log\log n)\}$, $LP_\infty = \{\Pi \in \mathbf{P}: T(n) \in O(S(n) \log n)\}$. In particular, LP_0 is the class of problems for which LPM is as efficient as the standard sequential model, while the problems in LP_∞ are the less adequate to be solved in the new model. We present several algorithms belonging to the above classes, and derive some relative lower bounds. Finally we discuss important relations between parallel and LPM algorithms.

1. INTRODUCTION

We introduce a new sequential model of computation called Logarithmic Pipelined Model (LPM), in which a RAM processor with uniform cost criterion (word model) [AHU74] has pipelined access to a memory of m cells in time logm. Although our present study is focused on the theoretical computational bounds in the new model, its practical motivation clearly emerges from the study of VLSI systems. In fact, the usual assumption that a memory can be accessed in constant time becomes unacceptable as m increases, due to the limited fan-in/fan-out of logic gates. An access time of logm is consistent with this limitation, and seems to be adequate for the incoming VLSI technology [MC80]. The relevance of different access times is discussed in [C85,S88].

A computational model similar to LPM has been studied in [AACS87], where a processor is connected to a hierarchy of memory banks, with logm access time to the higher bank. Our work differs from [AACS87], since we consider a single bank with access time logm, and we allow pipelining in the memory access channels. This leads to completely different results.

We assume that our RAM processor has a fixed number of internal registers. In the algorithm specification, we shall assume that the processor is connected to the memory with

---

[1] This work has been supported in part by MURST of Italy.

three channels C_A, C_R, C_W, to transfer addresses from RAM to memory (C_A), and data in reading (C_R) and writing (C_W) operations. The comunication time along the channels is logm. A write operation requires time logm, since the address and the datum are sent contemporarely, while a read operation obviously requires time 2 logm. The channels allow pipelining, therefore a queue of logm data can be present in a channel at any given time. We also assume that the size of a RAM program is independent of the input size n of the problem to be solved, hence the program is stored in the internal registers of the processor. We also assume that n is polynomially related to the memory size, that is $m=n^k$, and $O(\log m)=O(\log n)$. Therefore we will analyze our algorithms under the assumption that the access time is logn, without affecting the orders of magnitude.

In this paper we consider different complexity classes in LPM, always referring to problems in the class **P** (polynomial deterministic time for the standard sequential model). For a problem Π of size n, $\Pi \in \mathbf{P}$, let S(n) be the time required by the best sequential algorithm; and let T(n) be the time required by an algorithm in LPM. Clearly we have $T(n) \geq S(n) + c \log n$, where the term clogn accounts for the initial (or final) access to memory (if required). On the other hand we can always find an algorithm with $T(n) \leq c \log n\, S(n)$. To study the possible complexity relation between S(n) and T(n) we define the following classes:

$LP_0 = \{ \Pi \in \mathbf{P}\colon T(n) \in O(S(n)) \}$,

$LP_\infty = \{ \Pi \in \mathbf{P}\colon T(n) \in O(S(n) \log n) \}$.

LP_0 is the class of problems for which LPM is as efficient as the standard sequential model. The problems in LP_∞ are instead inadequate to be solved in LPM. For example, finding the sum of n elments stored sequentially in memory belongs to LP_0, since the accesses to all the elements can be pipelined ($T(n)=S(n)+\log n$). The same problem on a linked list is instead on LP_∞, since the address of an element is not known until the previous element has been read. Intuitively, LP_0 is the class of problems for which the memory addresses of the data are known $\geq 2 \log n$ steps before their use, except for a number s of cases, with $s \in O(S(n)/\log n)$. In LP_∞, s is of O(S(n)).

Between LP_0 and LP_∞ we can determine an infinite sequence of classes of increasing complexity. In particular we define here:

$LP_1 = \{ \Pi \in \mathbf{P}\colon T(n) \in O(S(n) \log\log n) \}$,

$LP_2 = \{ \Pi \in \mathbf{P}\colon T(n) \in O(S(n)\log n/\log\log n) \}$.

Examples of problems in LP_1 and LP_2 are given in the following. In section 2 we show several algorithms belonging to different classes of LPM, and in section 3 we discuss some of the relative lower bounds. Important relations between parallel and LPM algorithms are discussed in section 4.

2. UPPER BOUNDS

Several problems can be simply shown to belong to LP_0, such as finding the sum of all elements, or finding the maximum, in a set of n elements. In sequential computation these problems are solved in time O(n), while in LPM we only pay the extra time of $c \cdot \log n$ for initial access, and for storage of the last result. Similarly we have:

Proposition 1. Multiplication of two n×n matrices using only semiring operations +, ×, can be done in time $O(n^3)$ in LPM, hence the problem belongs to LP_0.
Proof. The best sequential algorithm takes time $O(n^3)$. The proposition follows from the fact that the addresses of input data are known a priori, hence these data can be read in pipeline. Moreover a constant amount of intermediate results is to be recalled at any step, hence these results can always be maintained in the processor for immediate use. []

Proposition 2. The Discrete Fourier Transform of a vector of n elements can be computed in time O(nlogn) in LPM, hence the problem belongs to LP_0.
Proof. The FFT sequential algorithm can be seen as working in logn consecutive phases, each of which computes n partial results independently [AHU74].The addresses of all data are known a priori. In LPM each phase can be realized reading the data in pipeline, and storing the partial results in pipeline, in time n + c·logn. The total time is then O(n logn). []

A deeper analisys is needed to prove that other problems belong to LP_0. In particular, we use a relationship between parallel and LPM algorithms, to be discussed in section 4. We have:

Theorem 1. Finding a minimal spanning tree in a graph of n vertices is in LP_0.
Proof. The LPM algorithm discussed in section 4 runs in time $O(n^2)$, as the best known sequential algorithm (Primm). []

Theorem 2. The execution of $r \geq n$ consecutive searches in a set of n elements is in LP_0.
Proof. The LPM algorithm discussed in section 4, which runs in time O(r logn). []

Let us now study upper bounds in other complexity classes LP_i, $i>0$. We have:

Theorem 3. A complete binary tree stored by pointers can be visited in time O(n·loglogn) in LPM, hence the problem is in LP_1.
Proof. Since the request of a node returns information of two pointers, the number of known pointers doubles every 2 logn steps. Starting from the root pointer, the queues in the channels C_A and C_R can be filled of pointers in 2logn·loglogn steps. At this point logn nodes have been inspected, belonging to the upper subtree H of height loglogn. The procedure is interrupted (it is no more possible to double the number of pointers since the channel queues are full), and all the pointers contained in C_R, pointing to sons of the leaves of H, are stored in pipeline in a queue Q in memory. The overall time at this point is 2 logn loglogn + logn ∈ O(logn·loglogn). This process is repeated for each subtree H' of height loglogn, whose root pointer is an element of Q, and the pointers to the sons of the leaves of H' are added to Q. Since there are O(n/logn) such subtrees, the whole process terminates in time O(n·loglogn). This obviously implies that the problem is in LP_1. []

For more complex linked structures we may employ the technique of retrieving pointers in pipeline, as done in the proof of theorem 3. Others constraints, however, may prevent the algorithm to benefit from accumulation of pointers. This is the case of graph searching, where a mark must be stored with each inspected node, thus introducing a logn delay for the nodes. In fact, we have been unable to put graph searching in a lower class than LP_∞.

Let us now discuss how to traverse a binary tree stored in vector form.

Algorithm TRAVERSAL. Traversing a complete binary tree A of n nodes, stored in a vector form, along a path from the root to a leaf.

Method. For semplicity A is stored in a vector V, in breadth first order, and the procedure consists of logn/loglogn phases. In each phase the logn nodes of a subtree H of height loglogn are read from memory, and a path h of height loglogn is examined in H. In the next phase, a new subtree H' is read, whose root is a son of the lowest (leaf) node of h. (In the first phase, the subtree has the same root of A). To compute the memory addresses of the nodes of H, recall that, if the root is in the location i of V, the other nodes are in locations 2i, 2i+1, 4i, 4i+1, 4i+2, 4i+3,...(in particular, the highest subtree occupies the locations 1 to logn). The rule (R) adopted to select the path h in H depends on the problem (see the following proof of theorem 4). []

Complexity of TRAVERSAL. Each phase of TRAVERSAL takes time 2logn for reading the root of H, plus logn to read the succeeding nodes in pipeline. Hence, the total time required by the algorithm is $O(\log^2 n/\log\log n)$. []

Theorem 4. Searching an element in an ordered set of n elements can be done in time $O(\log^2 n/\log\log n)$ in LPM, hence the problem is in LP_2.

Proof. Represent the set as a perfectly balanced binary search tree, stored in vector form as in algorithm TRAVERSAL. Then apply TRAVERSAL with the following rule R. Let *e* be the searched element, *u* be the last element currently found in h, and $l(u)$ be the location of *u* in V (at the beginning $l(u)=1$). If $e=u$ the element is found. If $e<u$ (or $e>u$), wait for the element u' such that $l(u')=2l(u)$ (or $l(u')=2l(u)+1$). The thesis follows from complexity of TRAVERSAL. []

Theorem 5. A set of n elements can be sorted in time $O(n \log^2 n/\log\log n)$ in LPM, hence the problem is in LP_2.

Proof. Assuming that the set is stored in a vector V, use the following version of HEAPSORT:

1) build a heap in V;
   for all j from n to 2 repeat the following steps 1) and 2):
2) read, and exchange, V(1) and V(j);
3) apply TRAVERSAL with the standard heap rule R, to reconstruct the heap
   between V(1) and V(j-1).

Step 1) can be done with the standard O(n) algorithm, which will require O(n logn) time in LPM. Steps 2) and 3) require time O(logn) and $O(\log^2 n/\log\log n)$ respectively, hence the overall time is $O(n \log^2 n/\log\log n)$. []

In section 4 we show that sorting is in LP_0, using the LPM transformation of a parallel algorithm given in [L84]. Therefore, the result of theorem 5 is not strict. However, the sorting algorithm reported above is simple, while the one derived from [L84] will be regarded as practically unfeasible. Another interesting problem in LPM is the following:

Circuit Value. Given a Boolean combinational circuit containing n AND, OR and NOT gates with unbounded fan-out, an arbitrary number $\leq 2n$ of input terminals, and one output terminal, compute the value of the output as a function of a given input set.

Circuit Value can be solved sequentially in time O(n), and is known to be log-space complete [KL86]. In LPM two parameters are significant, namely the *depth* d (number of *levels*) of the circuit, and the*width* w_i of each level i (number of gates and incident arcs in that level). The computation for any given level can be done on pipelined data, but the results must be stored in memory before being used as inputs to the next levels. The total time is then:

$T(n) \in O(\Sigma_{i=1,d} (w_i + 2\log n)) \in O((w' + \log n)\, d)$, where $w' = \max_i(w_i)$,

that is, Circuit Value may be put in different classes according to the values of w' and d.

Finally note that to establish upper bounds in LP_∞ is not interesting, since any problem in **P** can be immediately brought into that class.

3. LOWER BOUNDS

For a problem $\Pi \in$ **P**, let $B_S(n)$ be the tightest sequential lower bound known for Π, and let $B_L(n)$ be a lower bound in LPM. Obviously $B_S(n)$ is also a (non interesting) lower bound in LPM. Relevant lower bounds in LPM bring the problems in higher classes than LP_0. For example we have already seen that, if Π requires to scan a linked list of n elements, a lower bound $\Omega(n \log n)$ holds for $B_L(n)$. If $S(n) \in O(n)$, then $\Pi \in LP_\infty$, and $\Pi \notin LP_{i<\infty}$. A relevant lower bound can be derived for searching:

*Theorem 6.*Searching an element in a set of n elements requires time $\Omega(\log^2 n/\log\log n)$.

Proof. Assume that the given set B is available in sorted order (the lower bound thus derived obviously holds in general). It is easy to see that, in this case, a decision tree D for searching will include all the comparisons between the searched element and the elements of B, and each of these comparisons will occur exactly once. Hence, the height of D is $\Omega(\log n)$. Each comparison requires reading an element of B, which is available in the processor after a delay of $2\log n$ from the request. Starting from the root of D, we can read the elements appearing in the top subtree of height $\log(2\log n) = \log\log n+1$ before the first comparison can be made. (This policy clearly leads to the smallest lower bound). Without loss of generality assume that the subtree is visited in breadth-first order, and that each comparison indicates that the search must proceed on the right descendant. Refer to fig.1, where each node is labelled by the time when its element is requested, starting from time 0 at the root. In parentheses we report the time when the result of the comparison is available. At time $2\log n$ we have visited the subtree H_1, and we know the result of the first comparison. Making the stronger hypothetical assumption that, at this time, we can decide to proceed in the tree search from the node labelled $2\log n$ instead of 2, we must again visit the subtree H_2 completely, before the result of the comparison for this node is available. Therefore we must retrieve from memory at least all the elements pertaining to the nodes of the subtrees $H_1, H_2, \ldots, H_k$, with $k \in \Omega(\log n/\log\log n)$. These nodes are in total $k \cdot 2\log n$, hence $B_L(n) = \Omega(\log^2 n/\log\log n)$. []

Since the upper and lower bounds of theorems 4 and 6 are the same, searching $\in LP_2$, and $\notin LP_{i<2}$, that is the algorithm of theorem 4 is optimal.

An interesting problem for the study of lower bounds is again Circuit Value. Consider a circuit consisting of a cascade of n gates $G_1, ..., G_n$. Each gate G_i, i>a, is fed by the outputs of G_{i-a} and G_{i-1}, therefore the outputs of the gates $G_{i-a}, ..., G_{i-1}$ must be stored at each instant i. For $a \in O(1)$, these outputs can be temporarely stored in the processor, then the problem can be solved in time O(n). For *a* greater than a constant, such outputs must be stored in memory, and cannot be retrieved before 2logn time units. Therefore, starting from time 0, the output of G_{a+1} cannot be computed before time 2logn. This output (hence the input to G_{2a+1}) cannot be retrieved before 2logn additional time units, etc. We have $T(n) \geq (n/a)\cdot 2\log n$, therefore $B_L(n) = \Omega(n\cdot \log n/a)$, which takes different values depending on the value of *a*.

4. RELATIONSHIP BETWEEN LPM AND PARALLEL MODELS

Good algorithms in LPM have the property that the addresses of the data to be used at any given time are known in advance. This suggests a relationship between LPM and parallel computations. In fact, in a single parallel step executed by p processor, the addresses of all the data relative to the p operations are known in advance, and their new values are computed independently. We now show how to take advantage of this property, simulating parallel algorithms in LPM. A reverse simulation seems not to apply, because LPM does not require to know in advance the operations to be performed, while, in parallel computation, groups of p operations must be known at the same time.

The most general model of parallel computation is the PRAM, where a set of RAM processors access a common memory in constant time. This model is regarded as unfeasible, yet can be simulated on a bounded degree network to connect the processors with a slowdown of at least logp (e.g., see [AHMP87]). More restrict models include fixed connection networks of processing modules performing fixed operations (e.g., sorting networks like in [AKS83]). We will now show how a PRAM program requiring time t on p processors can be simulated in LPM with a memory of size m, in time O(p+logm)t. This leads to a *feasible* LPM algorithm, even though the PRAM model is idealistic.

Let $P_1, ..., P_p$ be the PRAM processors, and P be the LPM RAM processor. We consider p and t to be functions of n, with $p \in O(n^k)$, while the program π_i for each P_i has size independent of n. In the most general case all π_i are different, therefore the global PRAM program $\pi_1, ..., \pi_p$ takes space $O(n^k)$. Since P has fixed size, the PRAM program cannot be contained in (known to) P in the LPM simulation, and will be stored in the memory M. For each processor P_i, the memory will also contain the *program counter* PC_i , stored in cell M[i], and the contents of the internal registers of P_i (*description* of P_i, indicated by D_i), stored in cell M[p+i]. The memory map is shown in table 1. The simulation algorithm is the following.

Algorithm SIMUL. Simulation of a PRAM computation in LPM.

Method. The algorithm consists of t consecutive phases, which simulate the t steps of the PRAM computation. Let $\pi_{1,j}, ..., \pi_{p,j}$ be the instructions executed by $P_1, ..., P_p$ at step j. In the corresponding phase j of SIMUL, such instructions are read from M, and executed in a pipelined sequence. For each $\pi_{i,j}$ first the program counter PC_i is read. After 2logn time units PC_i is known, and then $\pi_{i,j}$ is fetched together with D_i. After 2logn additional time units the instruction is decoded in P, and the proper actions are taken (see table 2). They consist of updating PC_i to PC'_i, and sending a write request for PC'_i on channels C_A, C_W; updating D_i to D'_i (if required); sending the proper request on C_A, with the corresponding data on C_W. If $\pi_{i,j}$ is a "read from memory" instruction, 2logn additional time units are needed to transfer the datum in D_i. In conclusion, $\pi_{i,j}$ is executed in (up to) three steps of 2 time units each, with intermediate gaps of 2logn time units. The executions of all the $\pi_{i,j}$, $1 \leq i \leq p$, are interleaved in these gaps. []

Complexity of SIMUL. Each phase j of SIMUL requires time 6logn+6, up the end of the execution of instruction $\pi_{1,j}$. The subsequent p-1 instructions are then completed in additional 6(p-1) time units. Since the algorithm performs t phases, the total time required is O((p+logn)t). []

address	1 ...... p	p+1 .... 2p	2p+1 ....
contents	PC_1 ... PC_p	D_1 ...... D_p	π_1 ....π_p

Table 1. Contents of the memory M in the PRAM simulation.

Type of instruction	**Actions**	**Requests on C_A**	**Data on C_W**
internal computation	$PC'_i:=PC_i+1$	write in M[i]	PC'_i
	D'_i :=update (D_i)·	write in M[p+i]	D'_i
jump to x	PC'_i:=x	write in M[i]	PC'_i
write y in M[x]	$PC'_i:=PC_i+1$	write in M[i]	PC'_i
	–	write in M[x]	y
read fom M[x]	$PC'_i:=PC_i+1$	write in M[i]	PC'_i
	–	read M[x]	–
	(after 2logn time units:	write in M[p+i]	M[x])

Table 2. Different actions for instruction $\pi_{i,j}$.

Algorithm SIMUL can be easily riformulated, to simulate programs on bounded degrees networks of processors with local memory banks. Similarly, it can be extended to simulate the computation of a fixed interconnection network of p processing elements and depth t, in time $O((p+\log n)t)$.

The best parallel algorithms to be simulated in LPM are the ones which minimize the order $O((p+\log n)t)$. Not necessarily these algorithm are efficient in parallel (e.g. belong to NC). In particular, if p is $\Omega(\log n)$, the best algorithms minimize the product pt. Letting R(n) be the LPM simulation time of one such algorithm A, the interest of this simulation is clearly limited to the case $R(n) < S(n)\log n$. If this relation holds, the simulated version of A must be compared with other LPM sequential algorithms.

For example, consider the problem of finding the maximum in a set (which has already been located in LP_0). This problem admits a parallel algorithm with $p=n/\log n$, and $t=\log n$ [KL86]. The simulation in LPM leads to time $O((n/\log n+\log n)\log n)=O(n)$, as already found in section 2. A second example is given by the minimal spanning tree problem for a given graph of n vertices, which can be solved in $O(n\log n)$ parallel time with $O(n/\log n)$ processors [B80], hence in $O(n^2)$ time in LPM.

A more interesting result holds for sorting, if we apply the parallel algorithm of [L84], which uses $p\in O(n)$ processors to sort n words in time $t\in O(\log n)$. The LPM simulation attains time $O(n\log n)$ which is obviously optimal, locating sorting in LP_0. Note, however, that this algorithm is of mere theoretical interest, since the original one makes use of unfeasible AKS networks. (Our LPM algorithm of section 2 is quite simple, but it puts the problem in LP_2).

Another interesting problem is the execution of r search operations, $r\geq n$, in a set of cardinality n. This problem can be solved in parallel in a CREW PRAM model with $p=n$ processors, and total time $O(r/n \log n)$. r/n consecutive phases are executed. In each phase, each processor searches for one element in time $O(\log n)$. The simulated LPM algorithm takes time $O(r\log n)$, while the repeated application the search algorithm presented in section 2 would require time of $O(r \log^2 n/\log\log n)$.

While parallel upper bounds provide upper bounds to LPM problems (via SIMUL), lower bounds in LPM provide lower bounds in parallel models. For a lower bound $B_L(n)$ in LPM, we have:

Theorem 7. For any parallel algorithm requiring time t on p processors, we have:

$$(p+\log n)t \in \Omega(B_L(n)).$$

Proof. Follows from the simulation of parallel algorithms in LPM. []

For example, we have shown in theorem 6 that $B_L(n) \in \Omega(\log^2 n/\log\log n)$ for searching. Therefore, $(p+\log n)t \in \Omega(\log^2 n/\log\log n)$, that is $pt \in \Omega(\log^2 n/\log\log n)$ for $p\in \Omega(\log n)$, and $t \in \Omega(\log n/\log\log n)$ for $p\in O(\log n)$. A parallel searching algorithm that meets the above lower bound can be built along the same lines of the LPM searching algorithm of theorem 4 [LP89]. The time required by this algorithm is $O(\log n/\log p)$. For example, consider the cases $p = \log n$ and $p=n^\alpha$, $0<\alpha\leq 1$. If $p = \log n$ we have $t\in O(\log n/\log\log n)$, thus reaching the lower bound of theorem 8. If $p = n^\alpha$, $0<\alpha\leq 1$, we have $t \in O(1)$. This is the fastest parallel algorithm, but the lower bound of theorem 8 is not met.

5. CONCLUDING REMARKS

In this paper we have presented a computational model based on the assumption that the access to a memory of m cells must take time $\Omega(\log m)$. In a sense this has always been verified in the existing computers, where the memory decoder is essentially a tree of gates. However, in real systems the clock time tends to hide this phenomenon.

Access time of logm is consistent with VLSI theory, where a sort of underlying lemma states that m points cannot be reached in less than logm steps. Large synchronized VLSI systems assume such a time bound, and allow pipelining as a standard feature [MC80]. For this reason we consider LPM as a significant sequential model of computation for coming technologies.

REFERENCES

[AACS87] A. Aggrawal, B Alpern, A.K. Chandra and M. Snir. A model for hierarchical memory.*Proc. 19-th ACM STOC* (1987) 305-314.

[AHU74] A. Aho, J. Hopcroft and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1984.

[AHMP87] H. Alt, T. Hagerup, K. Melhorn and F.P. Preparata. Deterministic simulation of idealized parallel computers on more realistic ones. *SIAM J. Comp*. 16 (1987) 808-835.

[AKS83] M. Ajtai, J. Kolmos and E. Szemeredi. An O(NlogN) sorting network. *Proc. 15-th ACM STOC* (1983) 1-9.

[B80] J.L. Bentley. A parallel algorithm for constructing minimum spanning trees. *J. Algorithms* 1 (1980) 51-59.

[C85] S.A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control* 64 (1985) 2-22.

[MC80] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison-Wesley, Reading, MA, 1980.

[KL86] G.A.P. Kindervater and J.K. Lenstra. An introduction to parallelism in combinatorial optimization. *Discrete Applied Mathematics* 14 (1986) 135-156.

[L84] T. Leighton. Tight bounds on the complexity of parallel sorting. *Proc. 16-th ACM STOC* (1984) 71-80.

[LP89] F.Luccio and L. Pagli. A model of sequential computation based on a pipelined access to memory. Dip. Informatica, Univ. of Pisa, Tech. Rep. TR/89 (1989).

[S88] D.B. Skillicorn. A taxonomy for computer architectures. *Computer* 21 (1988) 46-57.

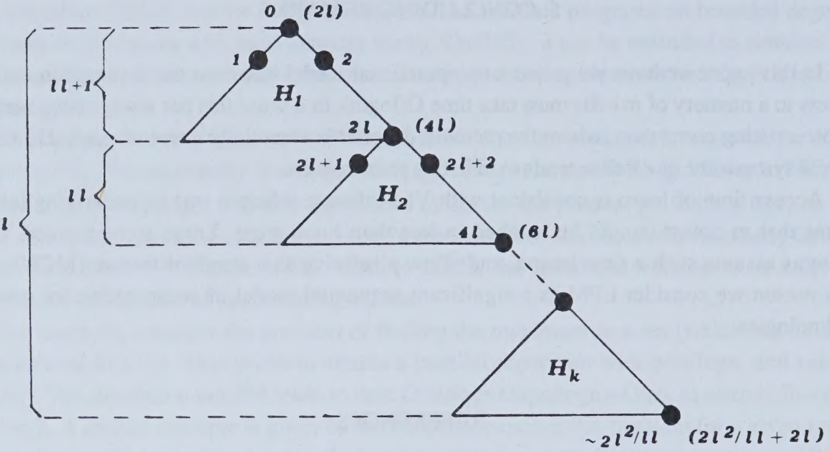

0
(2l)
1
2
H1
ll+1
2l
(4l)
2l+1
2l+2
H2
ll
l
4l
(6l)
Hk
~2l²/ll
(2l²/ll + 2l)

Fig.1. Decision tree for searching (l stands for logn).

NORMALIZED TIME AND ITS USE IN ARCHITECTURAL DESIGN

S. Ho, T. Holman,[1] L. Snyder
University of Washington,
Seattle, Washington

INTRODUCTION

Building better and faster computers is always the goal of computer design. To do this, designers often propose modifications and improvements to computers. Typically, these so-called improvements must also carry some cost, in additional size or complexity. All too often, only the benefits and not the costs are the subject of analysis. As an example, the Berkeley RISC design [Patterson] had a reduced instruction set, as well as register windows. The extra cost of the register windows was offset by the smaller control. But what then, if we had allocated this cost to, say, a carry-lookahead adder, or some other part? Would this have been a wiser choice?

Holman [1988] addressed this problem. The method of *normalized analysis* is a way of fairly resolving both the costs and benefits of a modification. [Holman 1989] A concrete example of such analysis is to ask:

> Do programs run faster on (parallel) computers when floating-point coprocessors are installed, or when the equivalent amount of hardware is used instead for additional processor elements?

We may repeat this question for each additional proposed modification, such as multipliers, shifters, *etc.*

This analysis allows determining whether a particular modification is, individually, cost-effective. In real designs, though, the number of potential modifications is not one, but many. Further, these changes may interact variously. A multiplier may obviate the need for a shifter, a shifter may duplicate part of a floating point unit, and so forth. We need an algorithm for taking the varied set of modifications, and choosing that set which, working in concert, provides the best cost-benefit ratio. We first extend the normalized analysis to the more understandable concept of *normalized time*. We then examine the effect of selecting multiple modifications with the simplest algorithm, the *greedy* algorithm.

MODEL

First, we must define our model. We start with some *base architecture*, and then evaluate the time and cost, on a fixed problem. The normalized time is then their product.

Definition 1 *Fix the computation. Then define*

$$\begin{aligned} T_0 &\equiv \textit{Time} \\ C_0 &\equiv \textit{Cost} \\ T_0C_0 &\equiv \textit{Normalized Time.} \end{aligned}$$

The subscript zero denotes the base architecture. The units, e.g. sm^2, are irrelevant, as we are only making comparisons here.

To the base architecture, we then add modifications. We stipulate that, akin to Amdahl's law [Amdahl], some fraction f is affected by the change, speeding it up by some factor S, and the rest is left undisturbed. We also stipulate that the change increases the cost by some fraction c.

As an example, a floating point coprocessor might produce a speedup of a factor of twelve, but only on sixteen percent of all instructions. It might also increase the cost, measured as chip area, by thirty-eight percent. (These figures are for relational operations in a bitonic sort on the Transputer T800. [Holman 1989])

Proposition 1 *A modification m affecting a fraction f, with speedup S and cost c obeys*

$$\begin{aligned} T &= T_0\left(1 - f + \frac{f}{S}\right) \\ C &= C_0(1 + c). \end{aligned}$$

The comparison is then between the normalized times. In our floating point example above, we find the normalized time is 1.18 times larger with the coprocessor than without. The coprocessor is not used enough, in the relational operations of this case, to be worthwhile, as the cost exceeds the benefit.

We can combine modifications by summing the time and cost. For simplicity, let us assume that the modifications do not interact. Interacting combinations would have a speedup term for each possible combination, but would otherwise be similar.

Proposition 2 *For a set of noninteracting modifications m_i, given f_i, c_i, S_i, we have*

$$\begin{aligned} T &= T_0\left[1 + \sum_i \left(-f_i + \frac{f_i}{S_i}\right)\right] \\ C &= C_0\left(1 + \sum_i c_i\right). \end{aligned}$$

TWO ARE BETTER THAN ONE

Before we consider the greedy algorithm, let us first examine the effect of the simplest combination: two noninteracting modifications combined. In this case, the cost is less than the product of the

---

[1] Currently with Sun Microsystems, Mountain View, CA

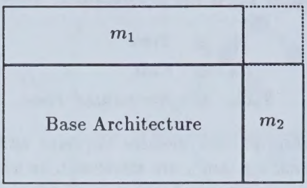


Figure 1: Representation of cross terms

```
algorithm greedy
$X \leftarrow \emptyset$
do
    for each $i$, $m_i \in M$
        compute $TC$
        if $TC > T_0C_0$ then $X \leftarrow X \cup \{m_i\}$
    $Base \leftarrow Base \cup X$
while $X \neq \emptyset$
```

Figure 2: The Greedy Algorithm

two costs, relative to the base, individually. This is expressed by the inequality

$$1 + c_1 + c_2 < (1 + c_1)(1 + c_2).$$

Graphically, this is demonstrated in Figure 1. The product overestimates the cost by the dashed interaction term. A similar relationship holds for the time. Thus, we have

Theorem 1 *The combined normalized time of two noninteracting modifications is less than the product of their separate normalized times.*

$$\frac{T_{12}C_{12}}{T_0C_0} \leq \frac{T_1C_1}{T_0C_0}\frac{T_2C_2}{T_0C_0}.$$

THE GREEDY ALGORITHM

Now we may consider the greedy algorithm, illustrated in Figure 2, for minimizing normalized time. The algorithm considers each modification in turn. All comparing favorably are added, becoming part of the new base architecture, and the process repeats until no (individually) favorable modifications remain.

Unfortunately, the greedy algorithm ignores the cross term described above.

Theorem 2 *The greedy algorithm is suboptimal.*

As an example, take $f_1 = f_2 = 1/2$, $c_1 = c_2 = 1$, and $S_1 = S_2 = 5$. We then have

$$\frac{T_1C_1}{T_0C_0} = \frac{T_2C_2}{T_0C_0} = \frac{6}{5}$$

$$\frac{T_{12}C_{12}}{T_0C_0} = \frac{3}{5}.$$

Neither m_1 nor m_2, taken alone, is worthwhile. The algorithm will leave the base architecture untouched, yet the optimal set is both of $\{m_1, m_2\}$.

Nevertheless, the greedy algorithm is conservative, in the sense that every greedily chosen modification is also a member of the optimal set. This is because the cross term is always positive.

Theorem 3 *The greedy algorithm is conservative.*

Proof: Let G be the greedily chosen set, and S the optimal set of modifications. Consider $G - S$. If nonempty, it must have normalized time less than one. Then, $S \cup (G - S)$ must have normalized time better than S, which is optimal, a contradiction. Therefore $G - S = \emptyset$, or $G \subseteq S$.

CONCLUSION

We began with the idea of normalized analysis: that the cost of a modification is just as important as its benefits. We have extended the model of normalized time to multiple groups of modifications. We then analyzed the results of the simplest, greedy, algorithm as a tool for selecting the best set of modifications.

In doing so, we find that the greedy algorithm is provably a suboptimal algorithm, even for the very simple types of modifications considered here. Nevertheless, since it is a conservative algorithm, it is still useful as a starting point for further selection. By running the fast and simple greedy algorithm, we can select many of the same modifications that would be found by any better algorithm, thus reducing the number of choices that the other algorithm must make.

With this theoretical basis, and the results of an initial algorithm, it may now be possible for computer designers to select, in a more analytical manner, which of the multitude of potential modifications to include in a computer system.

REFERENCES

Amdahl G. M. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proc AFIPS Vol. 30,* pp. 483-465, 1967.

Holman 1988 T. J. Holman, *Processor Element Architecture for Non-shared Memory Parallel Computers,* PhD Thesis, University of Washington, 1988.

Holman 1989 T. J. Holman and L. Snyder, "Architectural Tradeoffs in Parallel Computer Design," in *Decennial Caltech Conference on VLSI,* 1989.

Patterson D. A. Patterson and C. H. Sequin, "A VLSI RISC," *Computer,* 15(9):8-21, 1982.

A LOGIC FOR PETRI NETS AND ITS APPLICATIONS

HSU-CHUN YEN*

Abstract

In this paper, we define a first order logic for reasoning about Petri nets. We show that the satisfiability problem for this logic is EXPSPACE complete. Using this logic, a wide range of Petri net problems can be reduced to the satisfiability problem. Thus, this logic offers an umbrella under which many Petri net problems can be shown to be solvable in EXPSPACE.

1 Introduction

Petri nets provide an elegant and useful tool for modeling concurrent systems. In many applications, however, modeling by itself is of little practical use if one cannot analyze the modeled system. Despite the efforts made by many researchers, many analytical questions concerning Petri nets remain unanswered. In fact, the decidability of one of the most important questions concerning Petri nets, the reachability problem, was left unsolved for a long period of time until Mayr [19] finally provided an affirmative answer to the question just a few years ago. (The precise complexity of the reachability problem, however, has not yet been established.) Due to the degree of difficulty involved in many of the Petri net problems, it is desirable to have unified and systematic approaches to reason about Petri nets. Developing such a mechanism is exactly the goal of this paper. In this paper, we will define and investigate a first-order logic for Petri nets, and show that a number of results (some of which were previously unknown) can be obtained in a straightforward manner using this logic. In what follows, we first review some of the existing strategies known to be useful for examining the decidability and complexity of Petri net problems.

A useful tool for showing the decidability of Petri net problems is based on the Karp-Miller coverability graph analysis [16]. A coverability graph is a generalized reachability graph in which each potentially unbounded place is represented by a special symbol 'ω'. It has been shown in [16] that for any vector addition system (equivalently Petri net), its coverability graph is finite. As a result, a vector addition system (Petri net) is unbounded iff an ω occurs in its coverability graph. This technique has subsequently been used in a number of places for showing the decidability of Petri net problems. (See e.g., [6,7,26,27,28].) The key deficiency of the coverability graph approach lies in its inability to produce complexity bounds. This is because of the fact that the size of the coverability graph of a Petri net, in general, is not primitive recursive. As a result, the whole strategy is based on an unbounded search.

---

*Dept. of Computer Science, Iowa State University, Ames, IA 50011

Semilinearity has played a crucial role in the analysis of Petri nets. It is a common belief that the difficulties involved in obtaining results for general Petri nets lie in the fact that the reachability sets for Petri nets of dimension six or more are not in general semilinear [8]. However, they are restricted subclasses of Petri nets for which the reachability sets are semilinear. Utilizing results concerning semilinear sets and Presburger arithmetic, one can prove the decidability of the containment, equivalence, and reachability problems for a restricted subclass of Petri nets by demonstrating that the subclass exhibits semilinearity. In fact, a large body of results appearing in the literature rely exactly upon this. These include the decidability proofs of the containment, equivalence, and reachability problems for conflict-free [5], persistent [17,20,21], weakly persistent [31], normal [30], sinkless [30], 3-dimensional [29], 5-dimensional [8], symmetric [1], and regular [7,28] Petri nets. A drawback in using this approach is that the sizes of the semilinear set representations for some of the above classes were not known. Hence, no complexity bounds could be derived for those classes as a result of being semilinear. Recently, however, some efforts have been successfully made toward analyzing the size of the semilinear set representation for some of the above subclasses of Petri nets, thus yielding complexities for equivalence, containment, and reachability. (See e.g., [9,11,13,14].) A somewhat related approach, based on the notion of 'residue sets', has been used by Valk and Jantzen [27] to show the decidability for some Petri net problems. (See [27] for more details.) This approach, again, reveals no complexity bounds.

An important cornerstone in the complexity analysis of Petri nets is the EXPSPACE[1] upper bound of the boundedness problem shown by Rackoff [22]. This, together with an earlier result by Lipton [18] that the problem requires EXPSPACE, provides a near optimal solution for one of the most important Petri net problems. Rackoff's paper is significant not only because of the result itself, perhaps more importantly, it provides a technique for deriving complexities for other Petri net related problems. Such applications include the proofs of the complexity of the equivalence problem for commutative semigroups and symmetric vector addition systems (Huynh [14]), the equivalence between 2-way nondeterministic multihead finite automata and 2-way nondeterministic weak counter machines as well as a hierarchy theorem for 2-way nondeterministic weak counter machines (Chan [4]), a detailed multiparameter analysis of the boundedness problem for vector addition systems (Rosier and Yen [24]), and the complexity of the model checking problem for systems with many identical processes (German and Sistla [25]).

Since reachability and boundedness are known to be decidable, a natural way to show the decidability for an unknown problem is to reduce the unknown to reachability (or boundedness). In practice, however, this approach suffers from a drawback that many such reductions rely on rather complicated (and *ad hoc*) Petri net constructions which are hard to understand. This is perhaps because of the fact that the notion of boundedness (or reachability), as a reduction tool, is too weak to 'model' or 'simulate' other properties of Petri nets. In an attempt to avoid using 'brute force' reductions, a temporal logic for Petri nets was introduced ([12]) for which the model checking problem has been shown to be equivalent to reachability. In addition, a number of fairness related problems (fair nontermination problems, more precisely) concerning Petri nets have been reduced to the model checking problem. As a result, temporal logic provides an 'umbrella' under which certain fair nontermination problems can be shown to be decidable. (See [15] for a similar approach for showing the decidability of the nontermination problem with respect to finite-delay property.) In view of the above, the following question naturally arises: Is there a similar umbrella under which certain Petri net problems can be shown to be equivalent to boundedness?

---

[1] EXPSPACE=$\bigcup_{c>0} NSPACE(2^{n^c})$, where $NSPACE(f(n))$ is the class of languages accepted by nondeterministic Turing machines using at most $f(n)$ space.

As an attempt to answer the above question, in this paper we first define a first order logic for Petri nets. Each formula in our logic is of the form

$$\exists \mu_1, \mu_2, ..., \mu_k \exists \sigma_1, \sigma_2, ..., \sigma_k (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2 ... \xrightarrow{\sigma_k} \mu_k) \wedge F(\mu_1, ..., \mu_k, \sigma_1, ..., \sigma_k),$$

meaning that marking μ_i can be reached from μ_{i-1} $(1 \leq i \leq k)$ through the firing of transition sequence σ_i and predicate $F(\mu_1, ..., \mu_k, \sigma_1, ..., \sigma_k)$ holds. What makes this logic useful is that it is powerful enough to express many Petri net properties. For example, the unboundedness condition can be expressed as $\exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge (\mu_2 > \mu_1)$. By augmenting the proof of [22], we then show that if a formula is satisfiable, then there exists a short 'witness' (path) whose length is bounded by $O(2^{2^{c*n*logn}})$, for some constant c. As a result, the satisfiability problem can be solved in $O(2^{d*n*logn})$ space, for some constant d. Using our logic, we are able to provide simple and unified proofs for many known results, including the complexities of the boundedness, covering, and some fair nontermination problems, which would otherwise require more complicated arguments. More importantly, it also allows us to show EXPSPACE upper bounds for many unknown problems – the regularity detection problem (of [7,28]), for example. Another contribution of our result is that it explains why so many Petri net problems, even though bearing little similarity on the surface, possess the same complexity bound.

The remainder of this paper is organized as follows. In Section 2, we give the basic definitions of Petri nets and first order logic formulas. In Section 3, we show that the satisfiability problem for our logic is solvable in $O(2^{d*n*logn})$ space, for some constant d. In Section 4, we show that many Petri net problems, some of which were previously unsolved, can be reduced to the satisfiability problem. Thus, our main result may therefore be viewed as an umbrella under which the subsequent results in this paper are derived.

2 A logic for Petri nets

Let N (respectively, Z and R) be the set of nonnegative integers (respectively, integers and rational numbers), and let N^k (respectively, Z^k) be the set of vectors of k nonnegative integers (respectively, integers). A *Petri net* is a tuple (P, T, φ, μ_0), where P is a finite set of *places*, T is a finite set of *transitions*, φ is a *flow function* $\varphi : (P \times T) \cup (T \times P) \rightarrow \mathrm{N}$, and μ_0 is the *initial marking* $\mu_0 : P \rightarrow \mathrm{N}$. A *marking* is a mapping $\mu : P \rightarrow \mathrm{N}$. By establishing an order on P and T, i.e., $P = \{p_1, ..., p_k\}$ and $T = \{t_1, ..., t_r\}$, we can think of a marking μ as a vector in N^k, where the i-th component represents $\mu(p_i)$. (For convenience, $\mu(p_i)$ will be abbreviated as $\mu(i)$.) We also define the *transition vector* of a transition t, denoted by $\bar{t}$, to be a k-dimensional vector in Z^k such that $\bar{t}(i) = \varphi(t, p_i) - \varphi(p_i, t)$, and the set of transition vectors, denoted by $\bar{T}$, to be $\{\bar{t} \mid t \in T\}$. A transition $t \in T$ is *enabled* at a marking μ iff for every $p \in P$, $\varphi(p,t) \leq \mu(p)$. A transition t may *fire* at a marking μ if t is enabled at μ. We then write $\mu \xrightarrow{t} \mu'$, where $\mu'(p) = \mu(p) - \varphi(p,t) + \varphi(t,p)$ for all $p \in P$. A sequence of transitions $\sigma = t_1 ... t_n$ is a *firing sequence* from μ_0 iff $\mu_0 \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} \mu_n$ for some sequence of markings $\mu_1,...,\mu_n$. We also write $\mu_0 \xrightarrow{\sigma} \mu_n$. (We sometimes write $\mu_0 \xrightarrow{\sigma}$ if μ_n is not important.) A marking μ_n is said to be *reachable* from μ_0, denoted by $\mu_0 \xrightarrow{*} \mu_n$, iff $\exists \sigma$ such that $\mu_0 \xrightarrow{\sigma} \mu_n$. We let $R(P,T,\varphi,\mu_0) = \{\mu \mid \mu_0 \xrightarrow{*} \mu\}$ denote the set of all reachable markings. Let σ be a sequence of transitions. We define $\#_\sigma$ to be a mapping from T to N such that $\#_\sigma(t_i)$= the number of occurrences of t_i in σ. $\#_\sigma$ can be viewed as a vector in N^r whose i-th component is $\#_\sigma(t_i)$.

In what follows, we will present a first order logic for reasoning about Petri nets. In Section 3, the satisfiability problem will be shown to be solvable in EXPSPACE. Let (P,T,φ,μ_0) be a k-place r-transition Petri net. Our logic consists of the following elements:

1. *Variables:* There are two types of variables, namely, *marking variables* $\mu_1, \mu_2, ...$ and *path variables* $\sigma_1, \sigma_2, ...$, where each μ_i denotes a vector in Z^k and each σ_i denotes a finite sequence of transitions.
2. *Terms:* Terms are defined recursively as follows.
   (a) $\forall$ constant $c \in N^k$, c is a term.
   (b) $\forall\ j > i$, $\mu_j - \mu_i$ is a term, where μ_i and μ_j are marking variables.
   (c) $T_1 + T_2$ and $T_1 - T_2$ are terms if T_1 and T_2 are terms.
3. *Atomic predicates:* There are two types of atomic predicates in our logic, namely, *transition predicates* and *marking predicates.*
   (a) Transition predicates:
      - $y \odot \#_{\sigma_i} < c$, $y \odot \#_{\sigma_i} = c$ and $y \odot \#_{\sigma_i} > c$ are predicates, where $i > 1, y \in Z^r, c \in N$ and $\odot$[2] denotes the inner product.
      - $\#_{\sigma_1}(t_j) \leq c$ and $\#_{\sigma_1}(t_j) \geq c$ are predicates, where $c \in N$ and $t_j \in T$.
      - $\frac{\#_{\sigma_i}(t_j)}{\#_{\sigma_i}(t_l)} = z$, where $z \in R$, $i > 1$ and $t_j, t_l \in T$.

   (b) Marking predicates:
      - Type 1: $\mu(i) \geq c$ and $\mu(i) > c$ are predicates, where μ is a marking variable and $c (\in Z)$ is a constant.
      - Type 2: $T_1(i) = T_2(j), T_1(i) < T_2(j)$ and $T_1(i) > T_2(j)$ are predicates, where T_1, T_2 are terms and $1 \leq i, j \leq k$, meaning that the i-th component of T_1 equals, less than and greater than the j-th component of T_2, respectively.

   $F_1 \vee F_2$ and $F_1 \wedge F_2$ are predicates if F_1 and F_2 are predicates.

Let F be a predicate and D be a set of positive integers. We define $F^{[D]}$ to be the predicate resulting from removing type 1 atomic marking predicates of the form $\mu(i) \geq c$ and $\mu(i) > c$ from F, for all $i \notin D$.

In this paper, we deal with formulas of the following form (with respect to Petri net (P,T,φ,μ_0)):

$$\exists \mu_1, ..., \mu_m \exists \sigma_1, ..., \sigma_m \ (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \cdots \mu_{m-1} \xrightarrow{\sigma_m} \mu_m) \wedge F(\mu_1, ..., \mu_m, \sigma_1, ..., \sigma_m)$$

Given a Petri net $\mathcal{P}$ and a formula f, we use $\mathcal{P} \models f$ to denote that f is true in $\mathcal{P}$. The *satisfiability problem* is the problem of determining, given a Petri net $\mathcal{P}$ and a formula f, whether $\mathcal{P} \models f$.

---

[2] $(a_1, a_2, ..., a_k) \odot (b_1, b_2, ..., b_k) = \Sigma_{i=1}^{k} a_i * b_i$.

3 The main result

In this section, we show that the satisfiability problem is solvable in $O(2^{d*n*logn})$ space, for some constant d. In order to do so, we need a few lemmas first.

We begin by showing that predicates in our logic satisfy the so-called 'monotonic' property, which is essential in the proof of the main theorem.

Lemma 2.1: Let $\triangle$ ($\in Z^k$) be a vector whose first i components are nonnegative (i.e., $\forall j,\ 1 \leq j \leq i,\ \triangle(j) \geq 0$). For every predicate F, if $F(\mu_1, ..., \mu_m, \sigma_1, ..., \sigma_m)$=true, then $F^{[\{1,2,...,i\}]}(\mu_1 + \triangle, ..., \mu_m + \triangle, \sigma_1, ..., \sigma_m)$=true.

Proof. Omitted. □

The following lemma indicates that in deciding satisfiability, it is sufficient to consider formulas consisting of marking predicates only.

Lemma 2.2: Given a Petri net $\mathcal{P}= (P, T, \varphi, \mu_0)$ and a formula f, we can construct, in polynomial time, a Petri net $\mathcal{P}'= (P', T', \varphi', \mu_0')$ and a formula f' containing no transition predicates such that $\mathcal{P} \models f$ iff $\mathcal{P}' \models f'$.

Proof. Omitted. □

In what follows, we show that given a Petri net $\mathcal{P}= (P, T, \varphi, \mu_0)$ and a formula f, the length of the shortest path satisfying f, if exists, is bounded by $O(2^{2^{c*n*logn}})$, for some constant c. Before going into details, we require some definitions. Most of them are the same as in [22] (see also [24]). A finite sequence of vectors $w_1, w_2, ..., w_m \in Z^k$ is said to be a *path* in P if $w_1 = \mu_0$ and $w_{i+1} - w_i \in \bar{T}$ (the set of transition vectors), for all $i, 1 \leq i < m$. Let $w \in Z^k$ and $0 \leq i \leq k$. The vector w is i *bounded* if $w(j) \geq 0$ for $1 \leq j \leq i$. If $r \in N^+$ is such that $0 \leq w(j) < r$ for $1 \leq j \leq i$., the w is called *i-r bounded.* Given a predicate $F(\mu_1, \mu_2, ..., \mu_p)$, an i bounded ($i - r$ bounded) path $w_1, w_2, ..., w_m$ is called an i bounded ($i - r$ bounded) *F-path* if $\exists 1 \leq j_1, j_2, ..., j_p \leq m$ such that $F^{[\{1,2,...,i\}]}(w_{j_1}, w_{j_2}, ..., w_{j_p})$ is true. Let $m'(i, \mu, F)$ be the length of the shortest i bounded F-path whose initial marking is μ. (If no such path exists, then $m'(i, \mu, F) = 0$.) Let $g(i, F) = max\{m'(i, \mu, F) \mid \mu \in Z^k\}$.

To prove our upper bound result, we need the following lemma concerning the bounds of solutions of linear equations. The lemma is from [22]. (The proof is essentially from [3].)

Lemma 2.3: Let $d_1, d_2 \in N^+$, let B be a $d_1 \times d_2$ integer matrix and let b be a $d_1 \times 1$ matrix. Let $d \geq d_2$ be an upper bound on the absolute values of the integers in B and d. If there exists a vector $v \in N^{d_2}$ which is a solution to $Bv \geq b$, then for some constant c independent of d, d_1, d_2, there exists a vector $v \in N^{d_2}$ such that $Bv \geq b$ and $v(i) < d^{c*d_1}$ for all i, $1 \leq i \leq d_2$.

We are now in a position to prove our main result. The proof closely parallels the induction strategy used by Rackoff [22].

Lemma 2.4: If there is an $i - r$ bounded F-path in Petri net (P, T, φ, μ), then there is an $i - r$ bounded F-path of length $\leq r^{n^c}$, for some constant c independent of r and n.

Proof. Omitted. □

In what follows, we derive $g(i, F)$ recursively.

Lemma 2.5: $g(0, F) \leq 2^{n^c}$, for some constant c independent of n.

Proof. Omitted. □

Lemma 2.6: $g(i+1, F) \leq (2^n(g(i, F)+1))^{n^c}$ for all $i < k$, where c is a constant independent of n.

Proof. Omitted. □

Theorem 2.7: The satisfiability problem can be decided in $O(2^{d*n*logn})$ space, for some constant d independent of n.

Proof. Omitted. □

It is known that the boundedness problem for Petri nets is EXPSPACE-hard ([18]). Since our logic is powerful enough to express unboundedness, the following result follows:

Corollary 2.8: The satisfiability problem is EXPSPACE complete.

4 Some applications

In this section, we demonstrate the usefulness of Theorem 3.8 by first showing that many known complexity results concerning Petri nets follow immediately from our main result. We then derive some new results. Let $\mathcal{P}=(P,T,\varphi,\mu_0)$ be a k-place r-transition Petri net.

1. *Boundedness problem:*
   The boundedness problem is the problem of determining whether $R(P,T,\varphi,\mu_0)$ is finite. This problem has been shown to be solvable in EXPSPACE in [22]. (See also [24] for a detailed multiparameter analysis.) Using our logic, unboundedness can be formulated as $\exists\mu_1,\mu_2\exists\sigma_1,\sigma_2(\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2)\wedge(\mu_2>\mu_1)$.

2. *Coverability problem:*
   The coverability problem is to determine, given a vector $v\in N^k$, whether there exists a reachable marking μ such that $\mu\geq v$. This problem has also been shown to be solvable in EXPSPACE in [22]. Clearly, coverability can be formulated as $\exists\mu_1\exists\sigma_1(\mu_0 \xrightarrow{\sigma_1} \mu_1)\wedge(\mu_1\geq v)$ in our logic.

3. *(Strict) self-coverability problem:*
   In [14], the notions of *self-coverability* and *strict self-coverability* were introduced for solving the equivalence problem of commutative semigroups and symmetric vector addition systems. It was observed in [14] that the problem is solvable in EXPSPACE. Given a set of places I ($\subseteq P$), a path $\mu_0 \xrightarrow{*} \mu_1 \xrightarrow{*} \mu_2$ is *I-self-covering* if $\forall s\in I,\mu_2(s)\geq\mu_1(s)$ and $\forall s'\notin I,\mu_2(s')=\mu_1(s')$. It is strict if '$\geq$' in the above definition is replaced by '$>$'. The *(strict) self-coverability problem* is to determine, given an I, whether there is a (strict) I-self-covering path. Using our logic, the existence of an I-self-covering path can be expressed as $\exists\mu_1,\mu_2\exists\sigma_1,\sigma_2(\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2)\wedge((\bigwedge_{s\in I}\mu_2(s)\geq\mu_1(s))\wedge(\bigwedge_{s'\notin I}\mu_2(s')=\mu_1(s')))$. (For strict self-coverability, simply replace '$\geq$' by '$>$.)

4. *u-self-coverability problem:*
   This problem was also defined and solved in [14]. Given a $u\in N^k$, a path $\mu_0 \xrightarrow{*} \mu_1 \xrightarrow{*} \mu_2$ is *u-self-covering* if $\mu_2-\mu_1=u$. *u-self-coverability problem* is to determine, given a u, whether a u-self-covering path exists. This property can be expressed as $\exists\mu_1,\mu_2\exists\sigma_1,\sigma_2(\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2)\wedge(\mu_2-\mu_1=u)$.

5. *Final-state self-coverability problem:*
   This problem was defined and solved in [25] to study the complexity of the model

checking problem for a special type of systems with many identical processes. It was originally defined in the context of vector addition systems with states. Here we consider an equivalent version of the problem in the context of Petri nets. Let F be a set of places (corresponding to the set of final states in a vector addition system with states). A *final-state self-covering* path (with respect to F) is $\mu_0 \xrightarrow{*} \mu_1 \xrightarrow{*} \mu_2 \xrightarrow{*} \mu_3$ where $\mu_3 \geq \mu_1$ and $\exists s \in F, \mu_2(s) > 0$. The *final-state self-coverability problem* is to determine whether a such path exists. This problem can easily be formulated as $\exists \mu_1, \mu_2, \mu_3 \exists \sigma_1, \sigma_2, \sigma_3 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2 \xrightarrow{\sigma_3} \mu_3) \wedge ((\mu_3 \geq \mu_1) \wedge (\bigvee_{s \in F} \mu_2(s) > 0))$.

6. *Fair nontermination problems:*
   Let $\mathcal{A}$ be a finite set of nonempty subsets of transitions. Given an infinite sequence of transitions $\sigma = t_1, t_2, \ldots$, let $\inf^T(\sigma)$ be the set of transitions occurring infinitely often in σ. In [12] the following 6 types of fairness were defined: σ is said to be

   - T1-fair iff $\exists$ A $\in \mathcal{A}$, $\exists i \geq 1, t_i \in$ A.
   - T1′-fair iff $\exists$ A $\in \mathcal{A}$, $\forall i \geq 1, t_i \in$ A.
   - T2-fair iff $\exists$ A $\in \mathcal{A}$, $\inf^T(\sigma) \cap$ A $\neq \emptyset$.
   - T2′-fair iff $\exists$ A $\in \mathcal{A}$, $\inf^T(\sigma) \subseteq$ A.
   - T3-fair iff $\exists$ A $\in \mathcal{A}$, $\inf^T(\sigma) =$ A.
   - T3′-fair iff $\exists$ A $\in \mathcal{A}$, A $\subseteq \inf^T(\sigma)$.

   The *fair nontermination problem* with respect to T1 (T1′, T2, T2′, T3, T3′, respectively) fairness is the problem of determining whether a given Petri net has an infinite type T1 (T1′, T2, T2′, T3, T3′, respectively) fair computation. In [12], the nontermination problem was shown to be equivalent to the boundedness problem (and hence, solvable in exponential space) for all six types of fairness. In what follows, we show how to formulate the fair nontermination problem for all six notions of fairness using our logic. This will immediately yield the upper bounds. Given a subset of transitions A, let v_A be a vector in N^r such that $v_A(i)$=1 (0) iff $t_i \in (\notin)$ A. A Petri net is X-fair nonterminating iff

   - X=T1 $\iff \exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{A \in \mathcal{A}} \bigvee_{t_i \in A} 1 \leq \#_{\sigma_1}(t_i)) \wedge (\mu_2 \geq \mu_1))$.
   - X=T1′ $\iff \exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{A \in \mathcal{A}} \bigwedge_{t_i \notin A} \#_{\sigma_1}(t_i) \leq 0 \wedge \#_{\sigma_2}(t_i) \leq 0) \wedge (\mu_2 \geq \mu_1))$.
   - X=T2 $\iff \exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{A \in \mathcal{A}} \bigvee_{t_i \in A} 1 \leq \#_{\sigma_2}(t_i)) \wedge (\mu_2 \geq \mu_1))$.
   - X=T2′ $\iff \exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{A \in \mathcal{A}} \bigwedge_{t_i \notin A} \#_{\sigma_2}(t_i) \leq 0) \wedge (\mu_2 \geq \mu_1))$.
   - X=T3 $\iff \exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{A \in \mathcal{A}} (\bigwedge_{t_i \notin A} \#_{\sigma_2}(t_i) \leq 0) \wedge (v_A \leq \#_{\sigma_2})) \wedge (\mu_2 \geq \mu_1))$.
   - X=T3′ $\iff \exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{A \in \mathcal{A}} v_A \leq \#_{\sigma_2}) \wedge (\mu_2 \geq \mu_1))$.

In the remainder of this section, we show how to use Theorem 3.8 to derive new results. By reducing to the satisfiability problem, each of the following problems will be shown to be solvable in EXPSPACE.

1. *Regularity detection problem:*
   In [7,28], a subclass of Petri nets called *regular* Petri nets was defined. A Petri net is regular iff the set of all (finite) firable sequences of transitions defines a regular language (over T). An algorithm was given in [7,28] to determine whether a given Petri net is regular. A Petri net is not regular iff there exist markings $\mu_1, \mu_2, \mu_3, \mu_4$ such that μ_i, $1 \leq i \leq 4$, is reachable from μ_{i-1} and the following conditions are met (see page 314, [28]):

(a) $\mu_1 \leq \mu_2$ and $\mu_1 \neq \mu_2$ and

(b) $\forall p \in P, (\mu_1(p) \geq \mu_2(p)) \Rightarrow (\mu_3(p) \leq \mu_4(p))$, and

(c) $\exists p \in P, \mu_3(p) > \mu_4(p)$.

However, no complexity analysis was given regarding that particular algorithm. Using our logic, we have that a $k-$place Petri net is not regular iff $\exists \mu_1, \mu_2, \mu_3, \mu_4 \exists \sigma_1, \sigma_2, \sigma_3, \sigma_4 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2 \xrightarrow{\sigma_3} \mu_3 \xrightarrow{\sigma_4} \mu_4)$ and

(a) $(\mu_2 \geq \mu_1) \wedge (\bigvee_{i=1}^{k} \mu_2(i) > \mu_1(i))$,

(b) $\bigwedge_{i=1}^{k} (\mu_1(i) < \mu_2(i)) \vee (\mu_3(i) \leq \mu_4(i))$ and

(c) $\bigvee_{i=1}^{k} \mu_3(i) > \mu_4(i)$.

As a result, the regularity detection problem is solvable in exponential space.

2. *(Potential) determinism detection problem:*
   In [10], a subclass of Petri nets called *deterministic*[3] Petri nets was defined. A Petri net is deterministic iff for any reachable marking μ, there is at most one enabled transition in μ. It has also been shown in [10] that the boundedness, reachability, containment and equivalence problems can all be solved in EXPSPACE. Let v_t be the minimum vector for which transition t is enabled. Clearly, a Petri net is not deterministic iff $\exists \mu_1 \exists \sigma_1 (\mu_0 \xrightarrow{\sigma_1} \mu_1) \wedge (\bigvee_{t,t',t \neq t'} (\mu_1 \geq v_t) \wedge (\mu_1 \geq v_{t'}))$. A Petri net is *potentially deterministic* iff there does not exist an infinite path along which two or more transitions are enabled in infinitely many markings. (I.e., the Petri net can only be nondeterministic for a finite period of time.) It is fairly easy to see that a Petri net is not potentially deterministic iff $\exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigvee_{t,t',t \neq t'} (\mu_1 \geq v_t) \wedge (\mu_1 \geq v_{t'})) \wedge (\mu_2 \geq \mu_1))$. Consequently, the (potential) determinism detection problem can be solved in EXPSPACE.

3. *Frozen token detection problem:*
   The concept of *frozen token* was first introduced in [2] as a way to study the behavior of a concurrent system modelled by a Petri net. A Petri net has a frozen token iff there exists an infinite computation $\mu_0 \xrightarrow{*} \mu_1 \xrightarrow{t_{a_1}} \mu_2 \xrightarrow{t_{a_2}} \mu_3 \cdots$ and a place s such that $\forall i \geq 1$, $\mu_i(s) \geq 1 + \varphi(s, t_{a_i})$. (I.e., there exists a token that will never be removed from a place.) To detect a frozen token, we first modify the Petri net as follows. Let P be the set of places. Let p be a new place. For every $q \in P$, we introduce a transition t_q such that $\varphi(q, t_q) = \varphi(t_q, p) = 1$. Then the Petri net has a frozen token iff $\exists \mu_1, \mu_2 \exists \sigma_1, \sigma_2 (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge (\mu_1(p) > 0 \wedge \mu_2 \geq \mu_1)$. Thus, the frozen token detection problem is solvable in EXPSPACE.

4. *(Strong) promptness detection:*
   The concept of *(strong) promptness* was introduced in [27] as a way to deal with systems communicating with the environment. Let T_I and T_E be two disjoint sets of transitions such that $T_I \cup T_E = T$. (T_I and T_E can be viewed as the sets of internal and external transitions, respectively.) A Petri net (P, T, φ, μ_0) is said to be

---

[3] The reader should notice that the definition in [10] differs from the original definition of deterministic Petri nets given in [23].

(a) *Strongly prompt* (with respect to (T_I, T_E)) iff $\exists k \in N, \forall \mu \in R(P, T, \varphi, \mu_0), \forall w \in T_I^* : \mu \overset{w}{\rightarrow}\Longrightarrow \mid w \mid < k$, meaning that for every reachable marking μ, the longest sequence of internal transitions firable in μ is of length $< k$, for some k.

(b) *Prompt* (with respect to (T_I, T_E)) iff $\forall \mu \in R(P, T, \varphi, \mu_0), \exists k \in N, \forall w \in T_I^* : \mu \overset{w}{\rightarrow}\Longrightarrow \mid w \mid < k$, meaning that for every reachable marking μ, there is no infinite sequence of internal transitions firable in μ.

It is not hard to see that a Petri net is not prompt (strongly prompt) iff $\exists \mu_1, \mu_2 \; \exists \sigma_1, \sigma_2$ $(\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge ((\bigwedge_{t \notin T_I} \#_{\sigma_2}(t) \leq 0) \wedge (\mu_2 \geq \mu_1))$. As a result, the (strong) promptness detection problem is solvable in exponential space.

5. *y-synchronization problem:*

   In [26], the notion of *y-distance* was introduced. Given a Petri net $\mathcal{P}$ and a $y \in Z^r$, let $D(\mathcal{P},y) = sup_{\mu_0 \xrightarrow{\sigma}}(\mid y \odot \#_\sigma \mid)$. $\mathcal{P}$ is said to be *y-synchronized* if $D(\mathcal{P},y)$ is finite. The *y-synchronization problem* is that of determining, given a Petri net $\mathcal{P}$ and a y, whether $\mathcal{P}$ is y-synchronized. This problem has been shown to be decidable and EXPSPACE-hard in [26]. However, the precise complexity was left unanswered there. Now, we show that the problem is in fact EXPSPACE-complete. As mentioned in [26], $D(\mathcal{P},y)$ is finite iff for every path $\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2$, if $\mu_2 \geq \mu_1$, then $\mid y \odot \#_{\sigma_2} \mid = 0$. Using our logic, $\mathcal{P}$ is not y-synchronized iff $\exists \mu_1, \mu_2 \; \exists \sigma_1, \sigma_2 \; (\mu_0 \xrightarrow{\sigma_1} \mu_1 \xrightarrow{\sigma_2} \mu_2) \wedge (((y \odot \#_{\sigma_2}) > 0 \vee (y \odot \#_{\sigma_2} < 0)) \wedge (\mu_2 \geq \mu_1))$. As a consequence, the y-synchronization problem can be solved in EXPSPACE.

Acknowledgment: The author wishes to thank Professor L. Rosier for his suggestions and encouraging comments.

References

[1] Araki, T. and Kasami, T. Decidable problems on the strong connectivity of Petri net reachability sets. *Theoretical Computer Science*, 4:pp. 97–119, 1977.

[2] Best, E. and Merceron, A. Frozen tokens and d-continuity: A study in relating system properties to process properties. *LNCS 188*, pp. 48–61, 1985.

[3] Borosh, I. and Treybis, L. Bounds on positive integral solutions of linear diophantine equations. *Proc. AMS*, 55(2):pp. 299–304, March 1976.

[4] Chan, T. On two way weak counter machines. *Mathematical Systems Theory*, 20(1):pp. 31–42, 1987.

[5] Crespi-Reghizzi, S. and Mandrioli, D. A decidability theorem for a class of vector addition systems. *Information Processing Letters*, 3:pp. 78–80, 1975.

[6] Finkel, A. A generalization of the procedure of Karp and Miller to well structured transition systems. *ICALP*, pp. 499–508, 1987. To appear in *Information and Computation*.

[7] Ginzburg, A. and Yoeli, M. Vector addition systems and regular languages. *Journal of Computer and System Sciences*, 20:pp. 277–284, 1980.

[8] Hopcroft, J. and Pansiot, J. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:pp. 135–159, 1979.

[9] Howell, R. and Rosier, L. Completeness results for conflict-free vector replacement systems. *Journal of Computer and System Sciences*, 37:pp. 349–366, 1988.

[10] Howell, R. and Rosier, L. Personal communication. 1989.

[11] Howell, R., Rosier, L. and Yen, H. Normal and sinkless petri nets. *Fundamentals of Computation Theory*, 1989.

[12] Howell, R., Rosier, L. and Yen, H. A taxonomy of fairness and temporal logic problems for Petri nets. *13th Symposium on Mathematical Foundations of Computer Science*, 1988, to appear in *Theoretical Computer Science*.

[13] Howell, R., Rosier, L., Huynh, D. and Yen, H. Some complexity bounds for problems concerning finite and 2-dimensional vector addition systems. *Theoretical Computer Science*, 46:pp. 107–140, 1986.

[14] Huynh, D. The complexity of the equivalence problem for commutative semigroups and symmetric vector addition systems. *STOC*, pp. 405–412, 1985.

[15] Jancar, P. Decidability of weak fairness in Petri nets. *6th Symposium on Theoretical Aspects of Computer Science*, pp. 446–457, 1989.

[16] Karp, R. and Miller, R. Parallel program schemata. *Journal of Computer and System Sciences*, 3:pp. 147–195, 1969.

[17] Landweber, L. and Robertson, E. Properties of conflict-free and persistent Petri nets. *JACM*, 25:pp. 352–264, 1978.

[18] Lipton, R. The reachability problem requires exponential space. *Tech. Rep. 62, Yale University, Dept. of CS.*, 1976.

[19] Mayr, E. An algorithm for the general Petri net reachability problem. *SIAM J. Computing*, 13(3):pp. 441–460, 1984.

[20] Mayr, E. Persistence of vector replacement systems is decidable. *Acta Informatica*, 15:pp. 309–318, 1981.

[21] Muller, H. Decidability of reachability in persistent vector replacement systems. *Proceedings of the 9th Symp. on Mathematical Foundations of Computer Science*, pp. 426–438, 1980.

[22] Rackoff, C. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6:pp. 223–231, 1978.

[23] Ramchandani, C. Analysis of asynchronous concurrent systems by Petri nets. *Project MAC TR-120, MIT*, 1974.

[24] Rosier, L. and Yen, H. A multiparameter analysis of the boundedness problem for vector addition systems. *J. of Computer and System Sciences*, 32(1):pp. 105–135, 1986.

[25] Sistla, A. and German, S. Reasoning with many processes. *Proceedings of IEEE Symp. on Logic in Computer Science*, pp. 138–152, 1987.

[26] Suzuki, I. and Kasami, T. Three measures for synchronic dependence in Petri nets. *Acta Informatica*, 19:pp. 325–338, 1983.

[27] Valk, R. and Jantzen, M. The residue of vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21:pp. 643–674, 1985.

[28] Valk, R. and Vidal-Naquet, G. Petri nets and regular languages. *Journal of Computer and System Sciences*, 23:pp. 299–325, 1982.

[29] Van Leeuwen, J. A partial solution to the reachability problem for vector addition systems. *STOC*, pp. 303–309, 1974.

[30] Yamasaki, H. Normal Petri nets. *Theoretical Computer Science*, 31:pp. 307–315, 1984.

[31] Yamasaki, H. On weak persistency of Petri nets. *Information Processing Letters*, 13:pp. 94–97, 1981.

FINITE AUTOMATON APERIODICITY IS PSPACE-COMPLETE[†]

SANG CHO and DUNG T. HUYNH
Computer Science Program
University of Texas at Dallas
Richardson, Texas 75083

Abstract. In this paper, we solve an open problem raised by J. Stern "Is finite automaton aperiodicity **PSPACE**-complete?" by providing an affirmative answer.

In a paper entitled "Complexity of Some Problems from the Theory of Automata," Stern investigated the complexity of finite automaton aperiodicity. In that paper, one can find a polynomial space algorithm for the problem and a proof that the problem is **CoNP**-hard. Since there is a gap between the upper and lower bounds of finite automaton aperiodicity, the author raised question "Is finite automaton aperiodicity **PSPACE**-complete?" We will show that finite automaton aperiodicity is indeed **PSPACE**-complete. (The reader is assumed to be familiar with basic complexity-theoretic notions that can be found in [HU79].)

In the following we provide the necessary definitions. Given a finite alphabet Σ, the regular languages over Σ are those accepted by a finite state automata. Regular languages can be constructed from the finite sets of strings by Boolean operations (union and complement) together with concatenation and *-operation. *Star-free languages* are constructed like regular languages from the finite sets of strings but with the restriction that the *-operation is not allowed.

We now introduce the formal definitions of the problem.

Finite automaton aperiodicity is defined as follows:

Instance. A minimum state deterministic finite state automaton (DFA) M with input alphabet Σ.
Question. Does M recognize a star-free language?

We first introduce a condition that characterizes the star-free languages.

Proposition 1.a. [Sc65] A regular language $W \subseteq \Sigma^*$ is star-free iff W is aperiodic, i.e., for all element x of the syntactic monoid there is some integer n such that $x^{n+1} = x^n$.

Thus, a regular language W is not star-free iff some element x of the syntactic monoid has a nontrivial period, i.e., for all n, $x^{n+1} \neq x^n$. This condition can be stated in terms of minimum state DFAs as follows.

Proposition 1.b. A regular language accepted by a minimum state DFA M is not star-free iff there is a word $u \in \Sigma^*$ and a state p such that u defines a nontrivial cycle starting at p, i.e., (1) $\delta(p, u) \neq p$, and (2) For some positive integer r, $\delta(p, u^r) = p$.

[†] This research was partially supported by the National Science Foundation under Grant DCR-8696097 and by the Organized Research Awards Program of the University of Texas at Dallas

The following problem is the complement of finite automaton aperiodicity.

Definition 2. Finite Automaton Cycle Existence is defined as follows:

Instance. A minimum state DFA M with input alphabet Σ.
Question. Is there a word u of Σ^* that defines a nontrivial cycle for M?

Next we introduce a **PSPACE**-complete problem which we use to prove the **PSPACE**-hardness of finite automaton cycle existence.

Definition 3. Finite State Automata Intersection is the following problem:

Instance. A sequence $A_1, A_2, ..., A_n$ of DFAs having the same input alphabet Σ.
Question. Is there a string $x \in \Sigma^*$ accepted by each of A_i, $1 \le i \le n$?

It was shown in [Ko77] that finite state automata intersection is **PSPACE**-complete. Since the details of the construction will be needed later in our proof of the **PSPACE**-hardness of finite automaton cycle existence, we reproduce them here.

Lemma 4. [Ko77] Finite state automata intersection is **PSPACE**-complete.

Proof. It is easy to see that finite state automata intersection is in **PSPACE**. Next we reduce an arbitrary problem in **PSPACE** to finite state automata intersection. To this end, let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a single tape deterministic p(n) space bounded Turing machine where p is some fixed polynomial and $B \in \Gamma$ denotes the blank symbol. Let $x \in \Sigma^*$ be an input string and let $n = |x|$. Let $\Delta = (Q \cup \{\varepsilon\}) \times (\Sigma \cup \Gamma)$. A string $\mathrm{VALCOMP}_x = \#ID_0\#ID_1\#...\#ID_m\#\# \in (\Delta \cup \{\#\})^*$ represents a valid computation of M on input x if the following conditions are satisfied :

(1) Each ID_i is an instantaneous description of M consisting of M's tape content (padded out to length p(n) with B's), the position of M's head, and the state of M,

(2) Each ID_{i+1} follows from ID_i in one step according to the transition rules of M,

(3) ID_0 is the start configuration of M on input x, and ID_m is an accepting configuration.

Clearly, M accepts $x \in \Sigma^*$ iff there is a valid computation $\mathrm{VALCOMP}_x = \#ID_0\#ID_1\#...\#ID_m\#\# \in (\Delta \cup \{\#\})^*$ of M on input x. We can construct a collection of DFAs with input alphabet $\Delta \cup \{\#\}$ so that the intersection of the languages accepted by these DFAs will be the singleton set consisting of the string $\mathrm{VALCOMP}_x$ if it exists, and $\varnothing$ otherwise.

Without loss of generality assume that M always takes an even number of steps, and has a unique accepting state q_{acc}. Further M erases its tape before accepting and has the head at the left end of the tape in an accepting configuration. We construct a DFA A_{ID} which checks that each ID_i is indeed an instantaneous description.

Next we construct two groups of DFAs to check that each ID_{j+1} follows from ID_j in one step according to the transition rules of M. Recall that given the (i–1)st, ith and (i+1)st symbols of ID_j the ith symbol of ID_{j+1} can be determined from the transition rules of M. We construct a DFA A_i^{even} which accepts strings in sets of the form $(\#\Delta^{i-2} a_1 a_2 a_3 \Delta^{p(n)-i-1} \#\Delta^{i-2} b_1 b_2 b_3 \Delta^{p(n)-i-1})^* \#\#$ so that b_2 follow from $a_1 a_2 a_3$ according to the transition rule of M, where $a_k, b_k \in \Delta$, $1 \le k \le 3$. The structure of A_i^{even} is illustrated in Figure 1, where the states of A_i^{even} are numbered in such a way that the number assigned to a state indicates its "distance" from state s_i. Further, d_i denotes the dead state and f_i the final state of A_i^{even}. (Note that only states with the same distance can be equivalent.) From the simple structure of A_i^{even}, one can easily see that the minimum state DFA A_i^{even} can be constructed by a deterministic log-space bounded Turing machine.

Similarly, we construct a DFA A_i^{odd} which accepts strings in sets of the form $\#\Delta^{p(n)}(\#\Delta^{i-2} a_1 a_2 a_3 \Delta^{p(n)-i-1} \#\Delta^{i-2} b_1 b_2 b_3 \Delta^{p(n)-i-1})^* \#\Delta^{p(n)}\#\#$, so that b_2 follows from $a_1 a_2 a_3$ according to the transition rules of M. The structure of A_i^{odd} is illustrated in Figure 2, where the states are numbered in such a way that the number assigned to a state indicates its "distance" **mod** 2p(n)+1 from s_i.

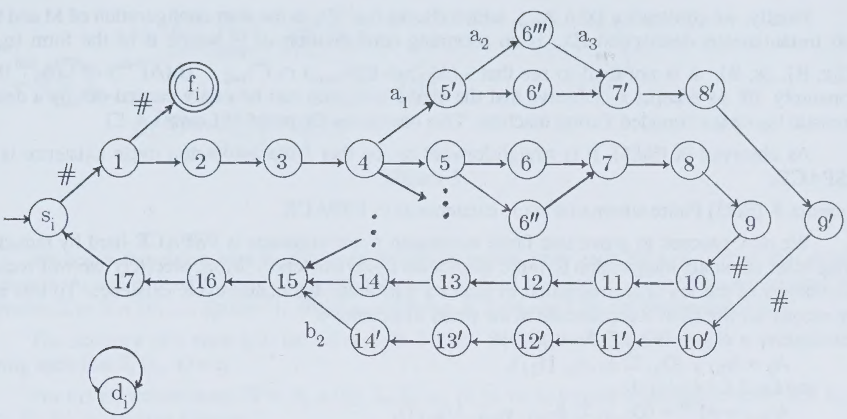

s_i
f_i
d_i
#
#
#
#
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
$5'$
$6'$
$7'$
$8'$
$9'$
$10'$
$11'$
$12'$
$13'$
$14'$
$6''$
$6'''$
a_1
a_2
a_3
b_2

Figure 1

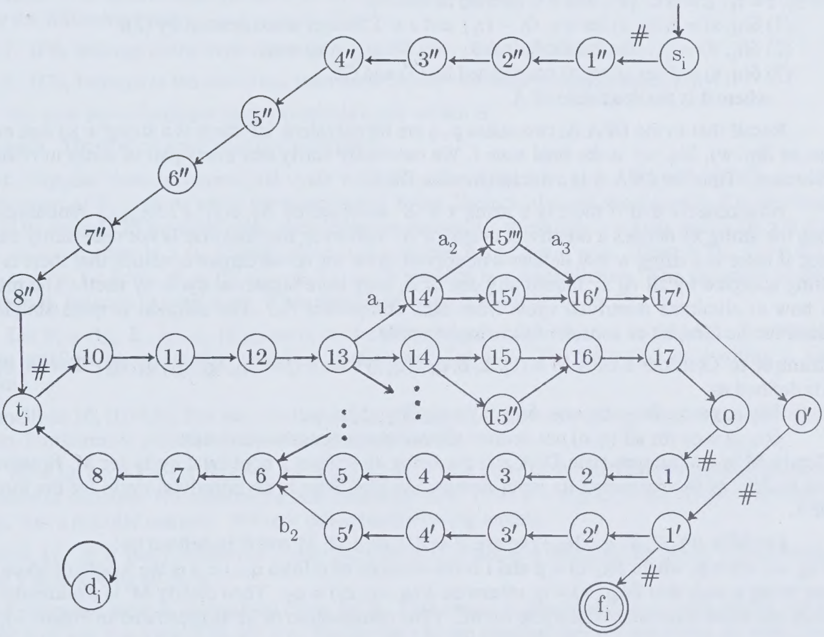

s_i
t_i
f_i
d_i
#
#
#
#
#
$1''$
$2''$
$3''$
$4''$
$5''$
$6''$
$7''$
$8''$
10
11
12
13
14
15
16
17
0
$0'$
1
2
3
4
5
6
7
8
$1'$
$2'$
$3'$
$4'$
$5'$
$14'$
$15'$
$16'$
$17'$
$15''$
$15'''$
a_1
a_2
a_3
b_2

Figure 2

Finally, we construct a DFA A_{ends} which checks that ID_0 is the start configuration of M and the last instantaneous description ID_m is an accepting configuration of M which is of the form $[q_{acc}, B][\varepsilon, B] \ldots [\varepsilon, B]$. It is not hard to see that $L(A_{ID}) \cap L(A_{\text{ends}}) \cap \bigcap_{i=2}^{p(n)-1} (L(A_i^{\text{even}}) \cap L(A_i^{\text{odd}}))$ is nonempty iff M accepts x. Observe that the above reduction can be easily carried out by a deterministic log-space bounded Turing machine. This completes the proof of Lemma 4. □

As observed in [St85], it is straightforward to see that finite automaton cycle existence is in **PSPACE**.

Lemma 5. [St85] Finite automaton cycle existence is in **PSPACE**.

We now proceed to prove that finite automaton cycle existence is **PSPACE**-hard by reducing finite state automata intersection to finite automaton cycle existence. More precisely we will reduce the outputs of the log space reduction of Lemma 4 to finite automaton cycle existence. To this end we reconsider the DFA's constructed in the proof of Lemma 4.

Let $A_1 = A_{\text{ends}} = (Q_1, \Sigma, \delta_1, s_1, \{f_1\})$,
$A_2 = A_{ID} = (Q_2, \Sigma, \delta_2, s_2, \{f_2\})$,
and for $2 \leq i \leq p(n)-1$:
$A_{2i-1} = A_i^{\text{odd}} = (Q_{2i-1}, \Sigma, \delta_{2i-1}, s_{2i-1}, \{f_{2i-1}\})$,
$A_{2i} = A_i^{\text{even}} = (Q_{2i}, \Sigma, \delta_{2i}, s_{2i}, \{f_{2i}\})$.

Without loss of generality, assume that $Q_i \cap Q_j = \varnothing$ if $i \neq j$ and all A_i's are minimum state DFAs.

We construct a DFA $A = (Q, \Sigma, \delta, s, \{f\})$ as follows:

$Q = \{d\} \cup \bigcup_{i=1}^{2p(n)-2} (Q_i - \{d_i\})$, where d_i is the unique dead state of A_i;
$s = s_1$, $f = f_1$, $\Sigma = \Delta \cup \{\#\}$, and δ is defined as follows:
(1) $\delta(q, a) = \delta_i(q, a)$ for $q \in Q_i - \{r_i\}$ and $a \in \Sigma$ except when defined by (2),
(2) $\delta(f_i, \#) = s_{i+1}$ for $1 \leq i \leq 2p(n)-3$, $\delta(f_{2p(n)-2}, \#) = s_1$,
(3) $\delta(q, a) = d$ for all (q, a) not defined by (1) and (2),
where d is the dead state of A.

Recall that in the DFA A, two states p, q are inequivalent iff there is a string w so that exactly one of $\delta(p, w)$, $\delta(q, w)$ is the final state f. We can easily verify that every pair of states in A are inequivalent. Thus the DFA A is a minimum state DFA.

Now observe that if there is a string $x \in \Sigma^*$ accepted by $A_1, A_2, \ldots, A_{2p(n)-2}$ simultaneously, then the string x# defines a nontrivial cycle for A. However, the converse is not necessarily true. In fact, if there is a string w that defines a nontrivial cycle for A, we cannot conclude that there is some string accepted by all A_i's. Indeed any one of A_i may have nontrivial cycle by itself. The problem is how to eliminate nontrivial cycle from each component A_i? The solution is quite simple. To illustrate the idea, let us consider following example.

Example 6. Consider a DFA $M = (Q, \Sigma, \delta, q_1, \{q_1\})$ where $Q = \{q_1, q_2, q_3, q_4, q_5\}$, $\Sigma = \{a, b\}$, and δ is defined as:
$\delta(q_1, a) = q_2$, $\delta(q_2, b) = q_3$, $\delta(q_3, a) = q_4$, $\delta(q_4, b) = q_1$,
$\delta(q, c) = q_5$ for all (q, c) not defined above, where q_5 is the dead state.

Clearly M is a minimum state DFA and the string ab defines a nontrivial cycle for M. However we can modify M by expanding its input alphabet so that there is no nontrivial cycle for the modified DFA.

Let $M' = (Q, \Sigma', \delta', q_1, \{q_1\})$ where $\Sigma' = \Sigma \times \{0, 1, 2, 3\}$ and δ' is defined by: $\delta'(q, <c, i>) = p$, where $\delta(q, c) = p$ and i is the distance of q from q_1, i.e. i is the length of some shortest string x such that $\delta(q_1, x) = q$; otherwise $\delta'(q, <c, i>) = q_5$. Then clearly M′ is a minimum state DFA and there is no nontrivial cycle for M′. (The construction of M′ is illustrated in Figure 3.) □

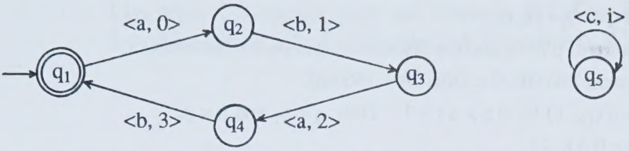


Figure 3

We apply the above idea to eliminate nontrivial cycles from each DFA A_i. Note that all the cycles in A_i are of length 2p(n)+2 except loops at the dead states d_i's. Therefore, we expand the alphabet Σ to $\Sigma \times \{0, \ldots, 2p(n)+1\}$. Before modifying A_i's we need the following definition.

The *distance* of a state q in the DFA A_i is defined to be $|x|$ **mod** 2p(n)+2 where x is a shortest string such that $\delta_i(s_i, x) = q$.

For the minimum state DFA $A_i = (Q_i, \Sigma, \delta_i, s_i, \{f_i\})$, we construct a minimum state DFA $B_i = (Q_i, \Sigma', \delta_i', s_i, \{f_i\})$ as follows:

$\Sigma' = \Sigma \times \{0, \ldots, 2p(n)+1\}$ and δ_i' is defined by :

$\delta_i'(q, <a, j>) = p$ if $\delta_i(q, a) = p$ and j is the distance of q,

Otherwise, $\delta_i'(q, <a, j>) = d_i$ where d_i is the dead state of B_i.

Before proving that there is no nontrivial cycle within B_i, for technical convenience, we want to classify B_i's into two classes. The first class contains exactly all B_i's with even i and is called the even class. The second class contains all B_i's with odd i and is called the odd class. We can easily verify the following facts.

Fact 7. If B_i belongs to the even class, then s_i is the only state with distance 0.

Fact 8. If B_i belongs to the odd class, then there is only one state t_i with distance p(n)+1.

We now show that there is no nontrivial cycle within B_i.

Lemma 9. There is no nontrivial cycle within B_i.

Proof. Suppose there is a nontrivial cycle within B_i. Let $\delta_i'(p, x) = q$, $p \neq q$ and $\delta_i'(p, x^r) = p$. By construction of B_i from A_i using the extension of Σ, the distance of p and the distance of q are identical. If B_i belongs to the even class, then there are strings x_1, x_2 so that $x = x_1x_2$ and $\delta_i'(p, x_1) = s_i$, $\delta_i'(s_i, x_2) = q$ since $|x| > 0$. Clearly $\delta_i'(q, x_1) = s_i$ and $\delta_i'(s_i, x_2x_1) = s_i$. Then $\delta_i'(p, x^r) = \delta_i'(p, x_1(x_2x_1)^{r-1}x_2) = \delta_i'(s_i, (x_2x_1)^{r-1}x_2) = \delta_i'(s_i, x_2) = q$ which is a contradiction. The argument is similar when B_i belongs to even class. Thus there is no nontrivial cycle for B_i. □

Let $B_i = (Q_i, \Sigma', \delta_i', s_i, \{f_i\})$ and d_i be the unique dead state of B_i, $1 \le i \le 2p(n)-2$. Let PRIME be the smallest prime number which is greater than 2p(n)–2. The following proposition is well known.

Proposition 10. [HW54] For any positive integer n there is at least one prime number p such that $n < p \le 2n$. Furthermore p can be computed in logn space. □

Now for each $1 \le i \le 2p(n)-2$, let $B_{2p(n)-2+i}$ be a new copy of B_i such that the sets of states are all pairwise disjoint. We construct a new minimum state DFA $B = (Q, \Sigma', \delta', s, \{f\})$ by concatenating B_i's in a circular manner. We now prove the following lemma.

Lemma 11. If B has a nontrivial cycle, then there is a string x accepted by all B_i's and hence a string y accepted by all A_i's.

Proof. Suppose there is a nontrivial cycle for B. By Lemma 9, this cannot be a cycle within any B_i. Let $\delta'(p, u) = q$, $p \neq q$ and $\delta'(p, u^r) = p$. Further let r be the smallest number satisfying the condition.

Let $p \in Q_i - \{d_i\}$ and $q \in Q_j - \{d_j\}$.

Claim 1. $i \neq j$ and the distance of p in B_i and the distance of q in B_j are the same.

Let DELTA = $(j - i)$ **mod** PRIME, $0 <$ DELTA $<$ PRIME.

Let $p_0 = p$ and $p_{k+1} = \delta'(p_k, u)$ for $0 \leq k \leq r - 1$. Thus, $q = p_1$ and $p = p_r$.

Let $p_k \in Q_{i_k} - \{r_{i_k}\}$ for $0 \leq k \leq r$.

Claim 2. For all $0 \leq k \leq r-1$, DELTA = $(i_{k+1} - i_k)$ **mod** PRIME.

Claim 3. r = PRIME.

Observe that DELTA is a generator of the cyclic group $\mathbf{Z}_{\mathrm{PRIME}}$. Therefore, the sequence $i_0, i_1, \ldots, i_{\mathrm{PRIME}-1}$ is a cyclic permutation of 1, 2, ... , PRIME. Now let u_1 be the shortest prefix of u such that $\delta'(p_k, u_1) = s_{i_k+1}$, and u_3 be the shortest suffix of u such that $\delta'(s_{i_{k+1}}, u_3) = p_{k+1}$. (See the following diagram.)

$$\rightarrow s_{i_k} \; - \; u_3 \rightarrow \; p_k \; - \; u_1 \rightarrow \; s_{i_k+1} \; - \; u_2 \rightarrow \; s_{i_{k+1}} \; - \; u_3 \rightarrow \; p_{k+1} \; - \; u_1 \rightarrow \; s_{i_{k+1}+1} \; \rightarrow$$

Then $u = u_1 u_2 u_3$ for some u_2 and it holds that $\delta'(s_{i_k+1}, u_2) = s_{i_{k+1}}$. Consider $u_3 u_1$. Clearly, $\delta'(s_i, u_3 u_1) = s_{i+1}$ for all $i = 1, \ldots, \mathrm{PRIME} - 1$. Further, $\delta'(s_{\mathrm{PRIME}}, u_3 u_1) = s_1$. Let x be such that $u_3 u_1 = x<\#, 2>$. Then x is accepted by all B_i's. Let x = $<a_1, 0><a_2, 1>\ldots<a_m, 2p(n)+1><\#, 0><\#, 1>$ and define $y = a_1 a_2 \ldots a_m \#\#$. Then clearly y is accepted by all A_i. This completes the proof of Lemma 11. □□

The remaining problem is that we have a variable size input alphabet instead of a fixed size alphabet. The idea is to encode such input symbols by binary strings of the same length of the form $1^{2k}00\{01, 10\}^k$, where k is the smallest integer that satisfy: the number of input alphabet $\leq 2^k$. Using some elementary argument we can show that there is a nontrivial cycle in DFA B with alphabet Σ iff there is a nontrivial cycle in DFA B′ with alphabet {0, 1}, where B′ can be obtained by replacing one transition of B by k transitions.

Thus we have proved the following lemma.

Lemma 13. Finite automaton cycle existence with input alphabet $\Sigma = \{0, 1\}$ is log-space complete for **PSPACE**. □

From Lemma 13, we obtain the following theorem as a corollary.

Theorem 14. Finite automaton aperiodicity is log-space complete for **PSPACE**. □

References

[HU79] Hopcroft, J. and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Co., 1979.

[HW54] Hardy, G.H. and E.M. Wright, *An Introduction to the Theory of Numbers,''* p.343 3rd ed., Oxford University Press, London, 1954.

[Ko77] Kozen, D., ''Lower bounds for natural proof systems,'' *Proc. 18th IEEE Annual Symp. on Foundations of Comp. Sci.*, pp.254-266, 1977.

[Sc65] Schützenberger, M.P., ''On finite monoids having only trivial subgroups,'' *Information and Control* 8, pp.190-194, 1965.

[St85] Stern J., ''Complexity of Some Problems from the Theory of Automata,'' *Information and Control* 66, pp.163-176, 1985

On the Complexity of Using BDDs for the Synthesis and Analysis of Boolean Circuits

Sreejit Chakravarty [1]
Dept. of Computer Science
State University of New York
Buffalo, NY 14260

ABSTRACT

We present the following results, pertinent to the use of BDDs in synthesizing and analysing Boolean circuit: (i) a characterization of Boolean circuits from which polynomial-sized BDDs can be computed; (ii) the containment problem for Simple BDDs is NP-Hard; (iii) a characterization for Boolean circuits for which the Satisfiability and Signal-Probability problems can be solved in polynomial time; (iv) the Satisfiability, Signal-Probability problems for 2-bounded BDDs is NP-Hard; and (v) a characterization of repeated-BDDs for which the two problems can be solved in polynomial time.

I. Introduction

Binary Decision Programs (**BDPs**)[18, 19] are a representation of Boolean functions that uses the notion of two-way branching. BDPs consist of a sequence of statements of the form $< label >:< variable >, < label1 >, < label2 >$. If the variable being tested has the value "true" then the program branches to *<label1>*, else it branches to *<label2>*. There are two special *"halt"* instructions labelled by Z and I. If the program halts at Z, it outputs *0*, else it outputs *1*. **BDDs** are labelled, directed graphs which are a diagramatic representation of BDPs. For every instruction I_0 in the BDP there exists a node N in the BDD ; if *<label1> (<label2>)* of I_0 is I_1 (I_2), then there exists an edge from N to the node for the instruction I_1 (I_2) labelled by *1 (0)*. The two nodes corresponding to the two halt instructions *Z,I* are the leaf nodes and are labelled by *0* and *1* respectively. Of interest here are BDDs that are acyclic. Figure 1 depicts some BDDs.

We next define some classes of BDDs. A variable X_i is said to **occur in a path** p if and only if there is a node labelled X_i in p. A **Free BDD** B is a BDD in which no variable occurs more than once in any path in B. A BDD that is not free is said to be a **Repeated BDD**. A Repeated BDD in which every variable occurs atmost k times in a path is said to be a **k-Bounded BDD**. Given a BDD B, define a binary relation $\leq_B$ on the set of input variables as follows: $\forall i \neq j, X_i \leq_B X_j$ if and only if $\exists$ a path p in B such that X_i occurs before X_j in p. A BDD B is an **Ordered BDD** if and only if B is a free BDD and $\leq_B$ is a partial order. The representations of [2,8,9] are Ordered BDDs. An ordered BDD in which every input variable labels atmost one node is a **simple BDD**.

The use of BDDs for the synthesis and analysis of VLSI circuits have been proposed by serveral authors. Synthesis of testable static CMOS circuits using free BDDs have been proposed in [12]. An approach to deriving a free BDD for F is to compute it from a Boolean circuit C_F for F. In this context, it is important to ascertain the size of B_F

---

[1] Research Supported by NSF Grant No. MIP-8802592.

in terms of the parameters of C_F. It is known[20] that for some Boolean functions the size of the smallest ordered BDD is exponential in the number of the input variables. On the other hand it is known that there exist Boolean functions for which there exist small BDDs[2]. However, from these facts we cannot ascertain the size of ordered BDDs that are computed from Boolean circuits. In this paper we present an upperbound on the size of BDDs computed from Boolean circuits. From this we deduce a characterization of Boolean circuits from which a polynomial-sized ordered BDD can be computed, in polynomial time. We are unaware of any such characterization of Boolean circuits.

The various analysis problems for which the use of ordered BDDs have been proposed include: Functional Testing[3,5]; Functional Design Verification[4,10]; Circuit Simulation[13]; and Compact Testing[6]. These applications require solutions to the Satisfiability, Signal-Probability and Equivalence problems. Here we are concerned with the complexity of these problems for various classes of BDDs. From these results we deduce some results for these problems for Boolean circuits.

We show that determining Satisfiability, Computing Signal Probability, for ordered BDDs can be solved in linear time. However, we show that the containment problem for ordered BDDs is NP-Hard. From this we deduce an upperbound for these problems for Boolean circuits. We next derive a characterization of Boolean circuits for which these problems can be solved in polynomial time. These results generalizes results in [17] for these problems for Boolean expressions.

Finally, we show that the Satsifiability and Signal Probability problems for repeated BDDs are NP-Hard and derive an upperbound for these problems for Repeated-BDDs. From this we deduce classes of repeated-BDDs for which there exists polynomial time solutions for these problems.

The Boolean circuit description we assume in this paper consists of two-input AND, OR and NOT gates. The results presented here can be extended for Boolean circuits consisting of NAND, NOR, AND, OR and NOT gates having bounded number of inputs. A primary input or the output of a gate is said to be a **fanout point** if and only if it drives the input of two or more gates. A Boolean circuit having no fanout point is said to be a **fanout-free circuit**. The proof of all the Lemmas and Theorems can be found in [21].

II. Fanout-Free Circuits

Procedure FANOUT-FREE is used for computing simple BDDs from fanout-free circuits. Boolean expressions and circuits in which only the inputs fanout (i.e. **Tree Circuits**) are equivalent. If this procedure is used for computing a BDD for a Tree circuit then we assume that every distinct occurrence of a variable is replaced by a new variable, so as to make it into a fanout free circuit. After constructing the BDD the reverse substitution is made.

FANOUT-FREE

Step-1. Construct the expression tree of the given tree-circuit or Boolean expression[1].
Step-2. Number the nodes of the tree using pre-order traversal. The root is numbered 0 and the numbers so assigned are the index of the node.
Step-3. Let N_i be the node with index i; N be the number of nodes in the expression tree; and g_i be a temporary Boolean variable associated with node N_i which denotes the Boolean function rooted at N_i. Traverse the expression tree in the same order as in step

2 and synthesize an attribute[1] for each node N_i as follows. The attribute computed for the rightmost leaf of the expression tree is the required BDD.

- Let the expression tree consist of just one node which is labelled by the variable X_t. The corresponding BDD is as shown in Figure 3(d).
- Let the expression tree consist of more than one node. Then, for all $0 \leq i < N$, a BDD B_i corresponding to the node N_i is computed as follows.

The root node of the expression tree can be one of the types shown in Figure 2(a)-(c), where the node indices are enclosed within (,). The corresponding B_0's are shown in Figure 3. In Figure 3, the variable g_i (g_j) is a temporary variable corresponding to the node numbered $i(j)$ in Figure 2.

For all $q \geq 0$, B_{q+1} is computed from B_q as follows. Let i be the least integer such that a node in B_q is labelled by g_i. Then, the node of B_q labelled by g_i is replaced as described below. 1. If the node N_i is a leaf node labelled by X_t then relabel the node labelled g_i by X_t.
2. If the node N_i is a non-leaf node then the occurence of the node labelled by g_i is as shown in Figure 4. The incoming edges are $e_1, \ldots, e_m$ and $n1, n2$ are its children. The node N_i can be one of the three types shown in Figure 5, where N_j, N_k are the children of N_i. The corresponding replacement rules are shown in Figure 6.

Lemma 2.1. If the input variable Y occurs k times in the Boolean expression E then the BDD B_E computed from E by procedure FANOUT-FREE has exactly k nodes labelled by Y.

Cor. 2.2. Procedure FANOUT-FREE correctly computes, in time $O(N)$, a simple BDD i.e. an ordered BDD of size $O(N)$ from a fanout-free circuit having N gates.

III. Circuits with Fanout Points

A fanout point R in a Boolean circuit C is a **head-fanout** point (**HFP**) iff R is such that there exists a path p from a primary input of C to R such that p does not pass through any other fanout point [15]. If a fanout point is not an HFP we say that it is a **dependent fanout** point (**DFP**). In Figure 7 A, B, C are HFPs and G is a DFP. Any given circuit can be decomposed into maximal tree sub-circuits such that the output of these sub-circuits are the fanout points. The tree sub-circuits of the circuit in Figure 7 are demarcated therein using dotted lines. The maximal tree-subcircuit whose output is the fanout point p is denoted by T_p.

Define the **fanout-graph** $G_C = (V, E)$ of the given circuit C as follows: V equals the set of HFPs; and for all $p, q \in V$, there exists a directed edge from p to q if and only if there exists a path r from p to q such that r does not pass through any other fanout point. The fanout graph of the circuit of Figure 7 is shown in Figure 8. Since the given circuit is acyclic therefore the fanout graph is also acyclic. Therefore, by topologically sorting the fanout graph we get a total ordering of the HFPs. The total ordering $< A, B, C >$ is derived from the fanout-graph of Figure 8.

An **assignment** is an ordered tuple $< F_1 = a_1, \ldots, F_f = a_f >$ where: $< F_1, \ldots, F_f >$ is a total ordering of the HFPs and for all $i, a_i \in \{0, 1\}$. An assignment $< F_1 = a_1, \ldots, F_f = a_f >$ is a **consistent assignment** (**CA**) if and only if there exists an input vector that sets F_i to a_i for all i. If an assignment is not consistent it is said to be an **inconsistent assignment** (**IA**). For the example of Figure 7, $< A = 0, B = 1, C = 1 >$ is a CA whereas $< A = 0, B = 0, C = 1 >$ is an IA.

Lemma 3.1. Let R be a DFP and F_R be the subset of HFPs such that for all $F_i \in F_R$ there exist a path from F_i to R. Then, an assignment of a consistent set of values to the members of F_R results in a unique value at R.

All the consistent assignments can be represented by an **assignment-tree**. In an assignment-tree all the non-leaf nodes are labelled by the names of HFPs and has two outgoing edges, one labelled by 0 and the other by 1. Also, for every CA there exists a path from the root to a leaf "labelled by" the CA. The assignment tree for the circuit of Figure 7 is shown in Figure 9. Note that the path $< N1, N2 >$ in Figure 9 is labelled by the CA $< A = 0, B = 1, C = 1 >$. Since, setting A to 0 uniquely determines the values of B and C there does not exist any node labelled by B or C along the path $< N1, N2 >$.

Our algorithm for computing ordered BDDs computes the assignment tree. It first assigns a constant to the minimal HFP and then extends this assignment to derive a CA. Enumeration of a path is completed if all the fanout points i.e. all the HFPs (see Lemma 3.1) has been assigned a constant. This is explained with the help of the example of Figure 7. Assume the total ordering of the HFP to be $S =< A, B, C >$. Since A is the minimal element of S it is the label of the root node. If A is set to 0, then B, C and G i.e. all the fanout points are set to a constant. Therefore, the path along this branch terminates at a leaf node. These constants can be determined by propagating the constant 0 from A using the following simple rules and breath-first traversal.

Rule 1. If any input to an AND (NAND) gate is 0 then the output of the gate is 0(1).
Rule 2. If any input to an OR (NOR) gate is 1 then the output of the gate is 1 (0).
Rule 3. If all the inputs to a gate are known then the constant at the output can ' determined.
If none of the above three rules can be applied then the constants cannot be propagated from the input to the output of the gate.

Continuing with our example, we next assign A the value 1. This constant cannot be propagated. Therefore, when A equals 1 B, the next unassigned HFP in the total order, can have the value 0 and 1. We assign the value 0 to B and propagate the constants. At the end of it C is left unassigned. We next assign the value 0 to C and propagate the constants. Thus, all the fanout points of the circuit has been assigned a constant and this completes the enumeration of the path $< N1, N3, N4, N6 >$. Enumeration of the rest of the paths in the assignment tree is done in a similiar fashion. The following lemma can be deduced from the informal description given above and observing that: (i) only consistent assignments are enumerated; (ii) the enumeration of each assignment takes time $O(m)$, where m equals the number of gates in the circuit; and (iii) the total number of consistent and inconsistent assignments equal 2^f.

Lemma 3.2. Let m be the number of gates in the circuit, f the number of HFPs and M the number of consistent assignments. Then, the assignment tree consisting of $O(M)$ nodes can be computed in time $O(M \times m)$ and $M \leq 2^f$.

From Lemma 3.1 we know that a consistent assignment implies a unique assignment to all the fanout points of the circuit. This in turn implies that once a consistent assignment is known all the inputs of the maximal tree circuits, that are branches of fanout points, are assigned a consistent set of constants. Given a tree subcircuit such that all its fanout inputs are assigned a constant it can be reduced to a fanout free circuit using the three rules stated above and Rule 4.

Rule 4. If there exists at least one input of a NAND (respectively, AND, NOR, OR) gate which is not assigned a constant and the rest of the inputs are assigned 1 (respectively, 1, 0, 0) then the inputs which have been assigned a constant are eliminated.

We next state the basic steps of our algorithm.

Procedure BC-BDD

Step 1. Determine the HFPs and a total ordering $< F_1, \ldots, F_f >$ of the HFPs of the given circuit.
Step 2. Compute the assignment-tree, using the above total order.
Step 3. Expand each nonleaf node of the assignment tree as follows. Let N_i labelled by F_j be the node being expanded; T_j be the maximal tree-subcircuit whose output is F_j; the path to N_i be labelled by the partial CA $< F_1 = a_1, \ldots, F_{i-1} = a_{i-1} >$; the edge $< N_i, M_0 >$ ($< N_i, M_1 >$) be labelled by 0(1); and e be the edge directed to N_i.
Step 3.1. From the partial CA determine the values of the primary inputs of T_j which is implied by this partial CA, using the four rules stated above.
Step 3.2. All the fanout points of T_j as well as some non-fanout points of T_j are assigned a constant at Step 3.1. Using these constants and Rules 1-4 reduce T_j to a fanout free circuit Q_j.
Step 3.3. Using procedure FANOUT-FREE compute the ordered BDD B_j from Q_j.
Step 3.4. Replace N_j by B_j as follows. The edges in B_j directed to the 0-node (1-node) are redirected to node M_0 (M_1) and the edge e now points to the root of B_j.
Step 4. Expand each leaf node as follows. Let T_y be the maximal tree circuit whose output is y. If some of the inputs of T_y are primary inputs then use Steps 3.1 - 3.4. Else, at the end of Step 3.2 T_y reduces to a constant a since all the inputs to T_y will be assigned a constant. The corresponding leaf node is replaced by a.

Let n = number of primary inputs; m = number of gates in the circuit; M = number of consistent assignments; and f = number of HFPs.
Lemma 3.3. BC-BDD computes an ordered BDD of size $O(M \times m) \leq O(2^f \times n)$ in time $O(M \times m) \leq O(2^f \times m)$.
The complete BDD computed by BC-BDD from the Boolean circuit of Figure 7 is shown in Figure 10. Theorem 3.4 gives us a characterization of Boolean circuits from which we can construct a polynomial-sized ordered BDD.
Theorem 3.4. If the given circuit C is such that $f \leq Log^k(n)$ for some constant k then there exists an ordered BDD B of size $O(n^{k+1})$ which can be computed in time $O(m^{k+1})$.

IV. Complexity of Problems for Boolean circuits, Ordered and Free BDDs

If R is a Boolean circuit (or an ordered BDD, or a free BDD) for a Boolean function F. Then the Satisfiability, Signal-Probability problems for Boolean circuits (ordered BDDs, free BDDs) is defined as follows.
Satisfiability. Given R determine an input vector, if it exists, that sets F to 1.
Signal-Probability. Given R and the probability distribution of the input variables, compute the probability that a randomly chosen input vector sets F to 1. Here the probability distribution of the input variables are assumed to be mutually independent.

Observe that if R is a free or ordered BDD then every path from the root of R to the 1-node defines an assignment of the input variables that sets F to 1. One such path can be computed in linear time. From this and Lemma 3.3 the following lemmas follows.
Lemma 4.1. The Satisfiability problem for free BDDs can be solved in linear time.
Lemma 4.2. The Satisfiability problem for Boolean circuit can be solved in time $O(2^f \times m)$, where f is the number of HFPs and m the number of gates in the given circuit.

A variable in a Boolean expression is said to be repeated if it occurs more than once in the Boolean expression. It was shown in [17] that an upperbound on the complexity of the Satisfiability problem for Boolean expressions is $O(2^r \times m)$ where r is the number of repeated variables in the Boolean expression and m is the size of the given expression. Since Boolean expressions are equivalent to tree-circuits and the repeated variables are equivalent to HFPs for this class of circuits Lemma 4.2 reduces to the result in [17] for Boolean expressions. The generality of Lemma 4.2 vis-a-vis the result in [17] follows from the observation that HFPs for Boolean circuits are a generalization of repeated variables for Boolean expressions.

We next have the following algorithm for computing signal probability of a Boolean function F from a free BDD B for F.

Algorithm Sig-Prob.

(* B is the given free BDD; $X_1, \ldots, X_n$ the input variables; for all $1 \leq i \leq n$, $p_i = \text{Prob}\{X_i = 1\}$ and $q_i = 1 - p_i$. For every node N we compute a probability P_N and $P_{root} = 1$. *)

- Visit the nodes of B breath-first and let N be the node being visited.
- Let $N_1, \ldots, N_t$ be the predecessors of N such that only one edge from N_i is incident on N; $M_1, \ldots, M_k$ the predecessors of N such that for all M_i both the outgoing edges of M_i are incident on N; X_j is the input variable labelling N_i; and the edge (N_i, N) is labelled by $a_i \in \{0, 1\}$.
- Define $r_i = P_{N_i} \times p_j$ if a_i equals 1; and $r_i = P_{N_i} \times q_j$ if a_i equals 0. Then, compute $P_N = \sum_{i=1}^{t} r_i + \sum_{i=1}^{k} P_{M_i}$.
- The probability computed for the 1-node is the required signal-probability.

Lemma 4.3. Signal-Probability for free BDDs can be computed using $O(N)$ arithmetic operations where N is the size of the BDD.

Lemma 4.4. Signal probability for a Boolean circuit C can be computed in time equal to $max(O(2^f \times n^3), O(2^f \times m))$.

Lemma 4.4 follows from Lemma 3.3, 4.3 and observing that n-bit arithmetic operations, which takes time $O(n^2)$, are enough to compute the signal probability. A better bound for Signal-Probability for Boolean circuits is discussed in [11]. Like Lemma 4.2, Lemma 4.4 is also a generalization of the result for Computing Signal Probability for Boolean expressions presented in [17]. Results similiar to Lemma 4.3 and 4.4 can be deduce for the Syndrome problem. We leave that to the interested reader.

Theorem 4.5. If in the given Boolean circuit C, $f \leq Log^k(n)$ for some constant k then the Satisfiability, Signal-Probability and Syndrome problem can be solved in polynomial time.

A Boolean function F_1 is said to be **contained** in F_2 iff for all input vectors v, $F_1(v) = 1 \Rightarrow F_2(v) = 1$. The **containment problem for simple BDDs** is defined as follows: Given two ordered BDDs B_1, B_2 representing the Boolean functions F_1, F_2 determine if F_1 is contained in F_2. Lemma 4.6 is from [17].

Lemma 4.6. The containment problem for fanout-free circuits is NP-Hard.

From Cor. 2.2 we know that given a fanout-free circuit C_1 (C_2) for F_1 (F_2) a simple BDD B_1 (B_2) can be computed in linear time. Thus, the containment problem for fanout-free expressions is embedded in the containment problem for simple BDDs. This leads to the next theorem.

Theorem 4.7. The containment problem for simple BDDs is NP-Hard.

This result is interesting because there exists polynomial time algorithms for most of the problems for ordered and simple BDDs, including some variations of the equivalence problems[7,14].

V. Complexity of problems for Repeated BDDs

Lemma 5.1 is from [17] and Lemma 5.2 follows from Lemma 2.1. Theorem 5.3 follows from Lemma 5.1 and 5.2.

Lemma 5.1. The Satisfiability, Signal Probability, Equivalence and Containment problems for 2-bounded Boolean expressions are NP-Hard.

Lemma 5.2. Procedure FANOUT-FREE computes a 2-bounded BDD from a 2-Bounded Boolean expression, in linear time.

Theorem 5.3. The Satisfiability, Signal-Probability, Syndrome, Equivalence and Containment problems for 2-bounded BDDs are NP-Hard.

The importance of the above theorem follows from the observation that a 1-bounded BDD is a free BDD and almost all the problems for free BDDs, except the Containment problem and the Equivalence problem[7], can be solved in polynomial time. In contrast, for 2-bounded BDDs all of the problems are NP-Hard. That is, like Boolean expressions bounded number of repetitions is enough to make the problems hard.

A variable X_i is said to be repeated in a repeated-BDD B iff there exists a path p in B such that X_i labels two or more nodes traversed by p. In Figure 1(a) X_1 is the repeated variable. Let $A =< X_1 = a_1, \ldots, X_r = a_r >$ be an assignment of constants to the repeated variables of B. The **pseudo-BDD** $B[A]$ is derived from B as follows: for all $1 \leq i \leq r$, if a node N is labelled by the input variable X_i then delete the outgoing edge of N labelled by $\overline{a_i}$.

Lemma 5.4. Let $B[A]$ be a pseudo-BDD with m nodes. Then (i) The Satisfiability Problem for $B[A]$ cab be solved in time $O(m)$; and (ii) The Signal Probability problem can be solved in time $O(m \times n^2)$.

Lemma 5.5. Let B be a repeated-BDD with r repeated variables, n the total number of input variables, and m the size of B. Then the Satisfiability problem for B can be solved in time $O(2^r \times m)$ and the Signal-Probability problem for B can be solved in time $O(2^r \times m \times n^2)$.

Note that the above argument cannot be used to derive upperbounds for the Equivalence and Containment problems for BDDs or Boolean circuits because of Theorem 4.7. The next theorem follows from Lemma 5.5.

Theorem 5.6. If a repeated-BDD B of size m is such that the number of repeated variables $r \leq (Log(m))^k$, for some constant k then the Satisfiability and Signal Probability problems can be solved in time $O(m^{k+1}), O(m^{k+1} \times n^2)$ respectively.

Acknowledgement I would like to thank Prof. H. B. Hunt III, Prof. D. J. Rosenkrant and Prof. S. S. Ravi for the discussion we had during the early stages of this research.

References

[1.] A. V. Aho and J. D. Ullman, *Principles of Compiler Design.* Addison-Wesley Publishing Company, 1979.

[2.] S. B. Akers, "Binary Decision Diagrams", *IEEE TC*, Vol. C - 27, No. 6.

[3.] S. B. Akers, "Functional testing using binary decision diagrams", *FTCS-8*, pp. 82 - 92.

[4.] S. B. Akers, "A procedure for functional design verification", *FTCS-10*, pp. 65 - 67.

[5.] M.S. Abadir and H.K. Reghabati, "Functional test generation for digital circuits described using binary decision diagram", *IEEE TC*, Vol. C - 35, No. 4.

[6.] B. Bhattacharya and S. C. Seth, "On the reconvergent structure of combinational circuits with applications to compact testing", *FTCS-17*, pp. 264 - 269.

[7.] M. Blum, A. K. Chandra and M. N. Wegman, "Equivalence of free Boolean graphs can be decided probabilistically in polynomial time", *Info. Processing Letters*, Vol. 10, No. 2, pp. 80 - 82.

[8.] R. Bryant, "Graph-Based algorithms for Boolean function manipulation",*IEEE TC*, Vol. C - 35, pp. 677 - 691.

[9.] R. Bryant, "Symbolic manipulation of boolean functions using a graphical representation", *Proc. 22nd Design Automation Conf.*, pp. 688 - 694.

[10.] R.Bryant, "Symbolic verification of MOS circuits", *Proc. 1985 Chapel Hill Conference on VLSI*, pp. 419 - 438. Computer Science Press.

[11.] S. Chakravarty and H. B. Hunt III, "On Computing Signal Probability and Detection Probability for Stuck-at Faults", To appear in *IEEE TC*.

[12.] S. Chakravarty, "A testable Multiplexer realization of CMOS combinational circuits", To be presented at *the 1989 IEEE Intl. Test Conf.*

[13.] E. Cerny and J. Gecsei, "Simulation of MOS circuits by Decision Diagrams", *IEEE CAD*, Vol. CAD-4, N0. 4.

[14.] S. Fortune, J. Hopcroft, E. M. Schmidt, "The Complexity of Equivalence and Containment for free variable program schemes", Lecture notes in Computer Science, No. 62, pp. 227 - 240. Ed. Goos, Hartmanis, Ausiello and Bohm. Pub. Springer-Verlag, Berlin.

[15.] H. Fujiwara and S. Toida, "The complexity of Fault Detection Problems for Combinational Logic Circuits", *IEEE TC*, Vol. C-31, pp. 555-560.

[16.] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, CA.

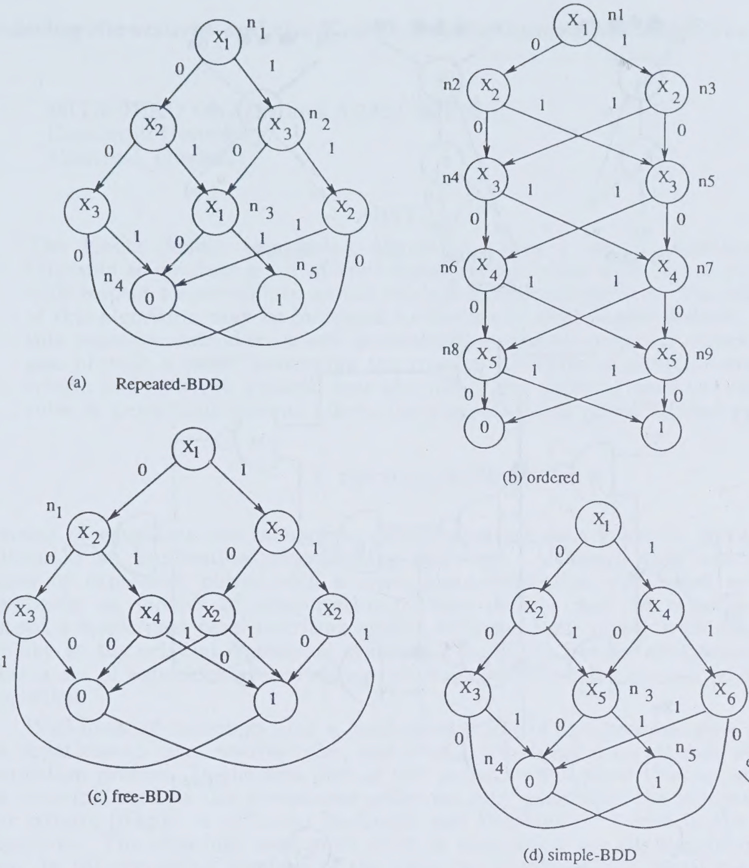
[17.] H. B. Hunt III and R. E. Stearns, "Monotone Boolean Formulae, distributive lattices, and the complexity of logics, algebraic structures, and computation structures", in *Proc. STACS86, 3rd Annu. Symp. Theoret. Aspects Comput. Sci.*, Vol. 210, pp. 277 - 291, 1986. Eds. B. Monien and G. Vidal-Naquet.

[18.] O. P. Kuznetsov, "Program realization of logical functions and automata Part 1: Analysis and Synthesis of Binary Processes", *Automata and Remote Control*, Vol. 38, pp. 1077 - 87, 1977.

[19.] C. Lee, "Representation of switching circuits by binary decision programs", *Bell Systems Tech. Journal*, Vol. 38, pp. 985 - 999.

[20.] J. Wegner, "Time-Space tradeoffs for branching programs", *Journal of Comp. and Syst. Sci.*, Vol. 32, No. 1.

[21.] S. Chakravarty, "On The Complexity of Using BDDs for the Synthesis and Analysis of Boolean Circuits". Tech. Rep. No. 89-12, Department of Computer Science, State University of New York, Buffalo, NY.


X1
X2
X3
X4
X5
X6
n1
n2
n3
n4
n5
n6
n7
n8
n9
0
1

(a) Repeated-BDD

(b) ordered

(c) free-BDD

(d) simple-BDD

Figure 1. Some example BDDs

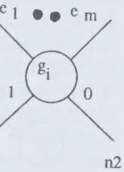

e1
em
gi
1
0
n1
n2

Figure 4

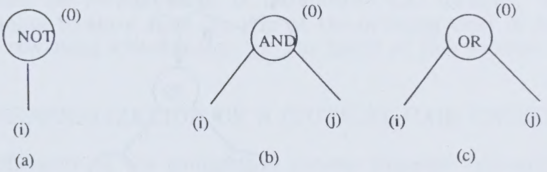

NOT
AND
OR
(0)
(i)
(j)
(a)
(b)
(c)

Figure 2

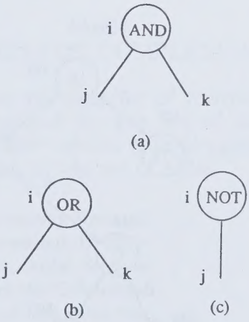
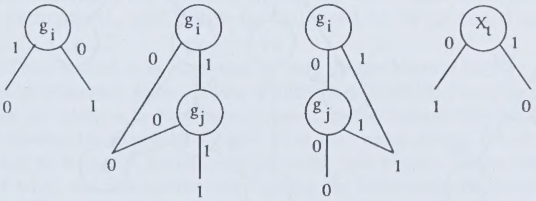

AND
OR
NOT
i
j
k
(a)
(b)
(c)

Figure 5



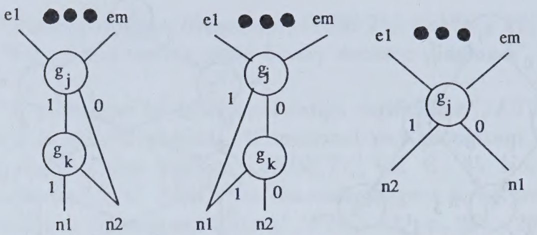
(a) NOT

(b) AND

(c) OR

(d)

Figure 3


e1
em
g_j
g_k
1
0
n1
n2

(a) AND (b) OR (c) NOT

Figure 6

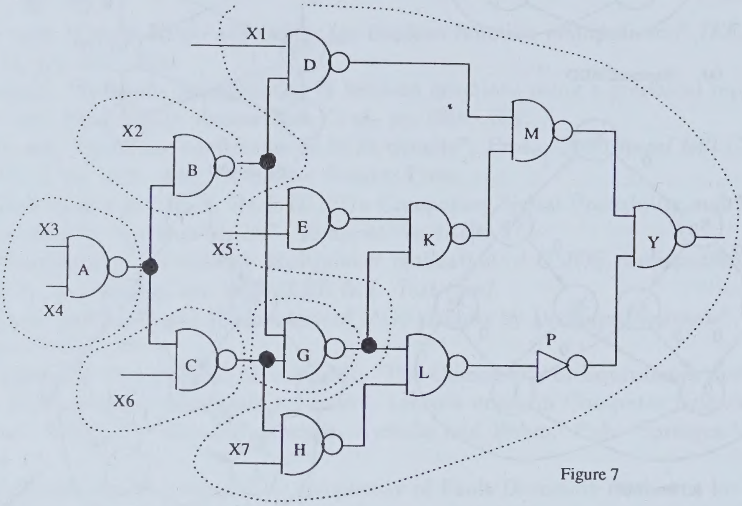

X1
D
X2
B
M
X3
A
X4
E
X5
K
Y
C
X6
G
L
P
X7
H

Figure 7

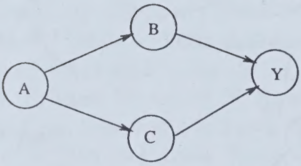
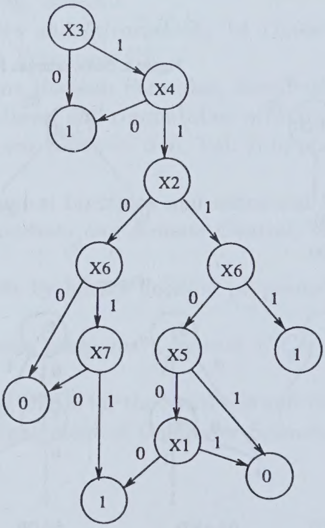


Figure 8


X3
X4
X2
X6
X6
X7
X5
X1
0
1

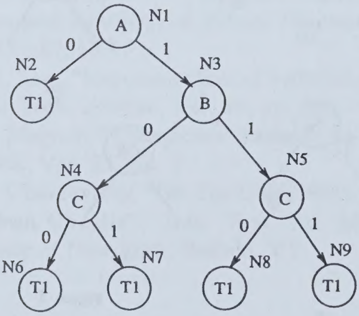

N1
A
N2
T1
N3
B
N4
C
N5
C
N6
N7
N8
N9
T1
0
1

Figure 9

Figure 10

Ordering Structures and the Knuth-Bendix Completion Algorithm†

MITSUHIRO OKADA and ADAM STEELE,
Concordia University,
Montreal, Quebec.

ABSTRACT

The Knuth-Bendix completion algorithm takes a set of equations and attempts to produce a set of convergent rewrite rules with the same power with respect to provability as the original equational system. The efficiency of this algorithm may be increased by the use of *critical-pair criteria*; and in this paper we introduce a new generalization of the *composite* criteria. We also provide a result concerning the relative strengths of *simplification* orderings; and show, in general, how the size of the ordering used to orient the rules in a rewriting system, affects the computational power of that system.

1. INTRODUCTION

Systems of equations can be used to specify abstract data types, or define programs written in an applicative programming language. Although they combine a high degree of expressive power with a clear, clean semantics, equational systems have drawbacks as models of computation. These defects may be remedied by using instead a convergent term rewriting system with the same power with respect to provability as the original equational system. The Knuth-Bendix completion algorithm takes a set of equations and attempts to produce a rewriting system with the above properties.

Well-founded orderings play a fundamental rôle in this process; both in orienting the equations to form rewrite rules, and also in providing a correctness proof for the completion process. In the first part of this paper we will show that by manipulating the ordering used in the correctness proof we may generalize the composite critical-pair criteria (Kapur et al.[8], and Bachmair and Dershowitz[3]) used in the completion algorithm. The orderings used most often in completion are the simplification orderings. In the remaining portions of the paper we provide a new result concerning the relationship between these orderings and certain proof theoretic ordinals, which supplements the former result of Dershowitz and Okada[4]. We also use proof theoretic arguments to show how the size of the ordinals used in completion affects the power of the rewriting systems that are the result of this process.

2. A GENERALIZATION OF A CRITICAL-PAIR CRITERION FOR COMPLETION

The efficiency of the completion process depends critically on the number of rewrite rules and critical-pairs generated. This number may be reduced by the use of so called critical-pair criteria which are tests that determine whether certain critical-pairs are redundant, and hence do not need to be oriented and added to the set of rules.

---

† This research was supported in part by the Natural Science and Engineering Research Council (Canada) under Grants OGP3663, ICR39976, Fonds pour la Formation de Chercheurs et l'Aide a la Recherche (Quebec) under Grants EQ41840, NC0025, Social Sciences and Humanities Research Council (Canada) under Grant 410-88-884, and the Committee on Aid to Research Activities (Concordia Univ.) under Grant N42. Part of this work was done when the first author was visiting the Laboratoire de Recherche en Informatique (L. R. I.), Universite de Paris-Sud, with support from Ministere de l'Enseignment National (France).

In this section we will introduce a new more general critical-pair criterion based on the notion of *compositeness* introduced by Kapur et al[8]. And following Bachmair and Dershowitz[3] we will prove the correctness of the new criterion, and of completion in general, by providing a well-founded ordering for the rewriting relation they induce on mixed equational and rewrite proofs. The following version of the Knuth-Bendix completion algorithm is taken from Huet[7], but the simplification rules have been adjusted to conform with those of Bachmair and Dershowitz[3], and our new more general test for composite critical pairs[15] has also been added. The initial data for the algorithm is: a set of equations E and a well-founded ordering $>$ which satisfies the properties of monotonicity and stability under substitution. Simplification orderings are those ordering structures that satisfy these two properties as well as the subterm property; and these orderings are the most commonly used orderings in actual implementations of the completion algorithm.

```
Let $E_0 := E$; $R_0 := \emptyset$; $i := 0$; $p := 0$.

    while there remain unmarked rules
      while $E_i \neq \emptyset$ do
        select an equation $s=t \in E_i$
        if $\hat{s} = \hat{t}$ then
          $E_{i+1} := E_i - \{s=t\}$; $i := i+1$
        else if $\hat{s}$ and $\hat{t}$ are comparable then
          if $\hat{s} > \hat{t}$ then $l := \hat{s}$; $r := \hat{t}$ else $l := \hat{t}$; $r := \hat{s}$
          let $K$ be the set of rules with label k whose left-hand sides
          are reducible by $l \rightarrow r$ to $l_k'$ and $l_k > l$ or $l_k \rhd l$
          $E_{i+1} := E_i - \{s=t\} \cup \{l_k' = r_k \mid k \in K\}$
          $p := p+1$
          $R_{i+1} := \{j : l_j \rightarrow r_j' \mid j \notin K\} \cup \{p : l \rightarrow r\}$
          where $r_j'$ is a normal form of $r_j$ with
          respect to rules from $R_i \cup \{l \rightarrow r\}$
          $i := i+1$
        else
          exit(failure - incomparable terms)
        endif
      endwhile
      Compute critical-pairs
        select an unmarked rule from with label $k$ from $R_i$
        $E_{i+1} := CP(k,R_i) - GCC(k,R_i)$
        where $CP(k,R_i)$ is the set of critical pairs between rule $k$
        and all rules with labels less than or equal to $k$ in $R_i$,
        and $GCC(k,R_i)$ is the set of all such critical pairs such
        that the critical overlap $s\sigma[l\sigma]$ between rules $s \rightarrow t$ and $l \rightarrow r$
        is reducible by rule $l' \rightarrow r'$ and either $l\sigma > l'\sigma'$, or $l\sigma = l'\sigma'$ and $l \rhd l'$
    endwhile
```

The Knuth-Bendix Completion Procedure

The original composite criterion introduced by Kapur et al[8]. (and called *PCP* by Bachmair and Dershowitz[3]) required that the left-hand side of the rule $l' \rightarrow r'$ match on to a proper subterm of $l\sigma$ in the critical overlap $s\sigma[l\sigma]$ generated by the rules $s \rightarrow t$ and $l \rightarrow r$. Our criterion *GCC* is more general requiring only that either $l\sigma > l'\sigma'$, or $l\sigma = l'\sigma'$ and $l \rhd l'$ (where $\rhd$ is the proper subsumption ordering, i.e. $s = t\sigma$ and $s \neq t$).

Kapur et al. gave an example that showed the need for their restriction on the definition of compositeness, but we will now show that *GCC* correctly handles this example.

Consider the following system of rewrite rules.

1. $g(f(x,x)) \rightarrow i(x)$
2. $f(x,a(y)) \rightarrow j(x)$
3. $f(a(x),y) \rightarrow j(a(x))$

Kapur et al. argued that if the restriction of $l'\sigma'$ being a proper subterm of $l\sigma$ was lifted the superposition between rule 1 and rule 3, $g(f(a(x),a(x)))$ would be considered composite; yet its critical pairs $g(j(a(x)))$ and $i(a(x))$ are equivalent but not joinable. Under *GCC* this superposition is not considered composite because the left-hand side of rule 3 is not greater than the left-hand side of rule 2 under the subsumption ordering.

We now consider the following examples which shows how our extension of the composite criterion, *GCC*, handles two cases not recognized by the composite criterion, *PCP*. The first example removes the requirement that $l'\sigma'$ be a proper subterm of $l\sigma$, where $l \rightarrow r$ and another rule $s \rightarrow t$ form a composite overlap. The second example will remove the restriction that $l'\sigma'$ occur at at a subposition of $l\sigma$.

Consider the following system of rewrite rules.

1. $h(f(a,y)) \rightarrow g(a,y)$
2. $f(a,b) \rightarrow b$
3. $f(x,b) \rightarrow c$

In this system we have a composite overlap between rule 1 and rule 2, and non-composite overlaps between rules 1 and 3, and rules 2 and 3. The critical pair between rule 1 and rule 2, $g(a,b)=h(b)$ can be considered to be superfluous because it will be joinable once we have oriented the critical pairs between rules 1 and 3, and rules 2 and 3. We would still arrive at the same final system if we had used the *PCP* criterion but the rule derived from the critical pair between rules 1 and 2 would have been added and then removed during the completion process.

Consider now the following system of rewrite rules.

1. $h(g(x),f(x)) \rightarrow h(x,b)$
2. $g(a) \rightarrow b$
3. $f(a) \rightarrow c$

We have (by our definition) a composite overlap between rule 1 and rule 2 (rule 3 matches on the overlap $h(g(a),f(a))$ and $f(a) < g(a)$). We can consider the critical pair $h(a,g)=h(b,f(a))$ to be superfluous because after one simplification it will be the same as the rule eventually derived from the critical pair between rules 1 and rule 3.

The proof of correctness for the Knuth-Bendix completion algorithm given by Bachmair, Dershowitz and Hsiang[2] formulates completion as an inference system on pairs of equations and rewrite rules. This in turn generates a rewriting relation on mixed equational and rewrite proofs that is shown to be well founded by embedding the relation into an ordering that consists of the multiset extension of a lexicographic combination of $>>$ the multiset extension of $>$ the term ordering, $>$ the term ordering, $\rhd$ the proper subsumption ordering, and $>$ the term ordering, where the objects measured are a multiset of the complexity measures of individual proof steps e.g. if $s \rightarrow_R t$ by the rule $l \rightarrow r$ then the measure of that proof step is $(\{s\},l\sigma,l,t)$.

The criterion GCC defines the following rewriting relation on proof schemas; note: we indicate the fact that the term $l'\sigma'$ need not be a proper subterm of $l\sigma$ (in fact it may occur anywhere in the term $s\sigma$) by writing the critical overlap as $s\sigma[l\sigma][l'\sigma']$.

$$t\sigma \leftarrow s\sigma[l\sigma] \rightarrow s\sigma[r\sigma/l\sigma] \;=>$$

$$t\sigma \leftarrow_R s\sigma[l\sigma][l'\sigma'] \rightarrow_R s\sigma[l\sigma][r'\sigma'/l'\sigma'] \leftarrow_R s\sigma[l\sigma][l'\sigma'] \rightarrow_R s\sigma[r\sigma/l\sigma]$$

We construct a well-founded ordering $>_{GCC}$ that contains both the rewriting relations for completion and GCC.

Let P be a proof $(s_0, \ldots, s_n)$, by $M(P)$ we denote the multiset $[g(s_0,s_1), \ldots, g(s_{n-1},s_n)]$ where $g(s_{i-1},s_i)$ is

$(s_{i-1}, l\sigma, l, s_i, l'\sigma', l')$ if $s_{i-1} \rightarrow_R s_i$ by $l \rightarrow r$ and $s_{i-1} \rightarrow_R s_{i-2}$ by $l' \rightarrow r'$
(where $l'\sigma'$, and l' are $-$ if $i=1$)

$(s_i, l\sigma, l, s_{i-1}, l'\sigma', l')$ if $s_{i-1} \leftarrow_R s_i$ by $l \rightarrow r$ and $s_i \rightarrow_R s_{i+1}$ by $l' \rightarrow r'$
(where $l'\sigma'$, and l' are $-$ if $i=n$)

$(s_{i-1}\# s_i, -, -, -, -, -)$ if $s_{i-1} \leftrightarrow_E s_i$

The first four components are consistent with the complexity measure for completion and the last two reflect the amount of overlap between neighbouring rewrite steps. We let the ordering $>_{GCC}$ be the multiset extension of the lexicographic combination of the multiset extension $\gg$ of the ordering $>$, the ordering $>$, the proper subsumption ordering $\rhd$, the ordering $>$, the ordering $>$ again, an finally the proper subsumption ordering $\rhd$. the ordering $>$. $P > P'$ if and only if $M(P) >_{GCC} M(P')$.

Let P be the overlap $t\sigma \leftarrow_R s\sigma[l\sigma] \rightarrow_R s\sigma[r\sigma/l\sigma]$, and let P' be equal to $t\sigma \leftarrow_R s\sigma[l\sigma][l'\sigma'] \rightarrow_R s\sigma[l\sigma][r'\sigma'/l'\sigma'] \leftarrow_R s\sigma[l\sigma][l'\sigma'] \rightarrow_R s\sigma[r\sigma/l\sigma]$ where $l\sigma > l'\sigma'$, or $l\sigma = l'\sigma'$ and $l \rhd l'$. The two proof steps $s\sigma[l\sigma][l'\sigma'] \rightarrow s\sigma[l\sigma][r'\sigma'/l'\sigma']$ are both smaller than $s\sigma[l\sigma] \rightarrow t\sigma$ and $s\sigma[l\sigma] \rightarrow s\sigma[r\sigma/l\sigma]$ because they are smaller in the second and third positions; but if we just consider the first four components the we cannot have $P > P'$ since both proofs contain $s\sigma[l\sigma] \rightarrow t\sigma$ and $s\sigma[l\sigma] \rightarrow s\sigma[r\sigma]$. The ordering GCC correctly handles this problem since when $l\sigma$ is greater than $l'\sigma'$ the fifth component is larger in P than P', when $l\sigma$ and $l'\sigma'$ are equal the requirement that l be greater than l' under the subsumption ordering makes the sixth component larger. For details of the use of the above ordering to prove the correctness of the criterion GCC we refer the reader to the thesis of the second author[15].

In trying to decide whether using a critical-pair criterion will make the completion algorithm more efficient in practice we have to consider a number of factors. In general, a critical-pair criterion will reduce the number of critical-pairs generated, and hence the overhead required to reduce the critical-pair to its normal form, orient the reduced pair, and deal with any new pairs generated by the rule before it may eventually be removed by simplification; these costs being balanced against the cost of the criterion test. This is also a factor in deciding between using the criteria GCC or PCP. Although, more general GCC requires the use of the term ordering $>$ which is more expensive to calculate (probably super-exponential) than the proper subterm ordering used by the test PCP.

In certain cases (especially when dealing with systems of equations that have AC axioms) the reduction in critical-pairs from using a criterion can make the difference between success and failure of the completion algorithm. In this case the added cost is obviously worth it. However, critical-pair criteria may not always be useful. If we are considering completion as a partial algorithm for theorem proving in systems of equations that do not produce a finite set of rewrite rules, it is probably better to orient a composite critical-pair, rather than wait for a more general rule that would take its place.

3. PROPERTIES OF COMPLETION ORDERINGS

Various well-founded abstract orderings have been used to orient equations in Knuth-Bendix completion. One common approach has been to use ordering structures in the class of simplification orderings. In our previous paper[4] we showed that most of the ordering structures of this type, including the recursive path ordering (Dershowitz), the lexicographic path ordering (Kamin-Levy), versions of the semantic path ordering (Plaisted), the path ordering (Kapur et al.), the recursive decomposition ordering (Jouannaud-Lescanne) and others, are simple substructures of Ackermann's system of ordinal numbers from Proof Theory in Logic.

Since Ackermann's system has the properties of a simplification ordering[4] our former result suggests that Ackermann's system is a very general simplification ordering, and it has been conjectured that this system is the most general simplification ordering. In this section we will provide an affirmative result for this conjecture, and we will show that any simplification ordering is an initial segment of Ackermann's system. This implies that the notion of simplification ordering does not go beyond the limit of Ackermann's system. In other words if our ordering has the monotone property, stability under substitutions and the subterm property, then it cannot go beyond this limit. Since only the first two properties are required in the termination proofs (and in the proof of correctness of the completion algorithm), in order to make the ordering stronger what we can do is to relax the subterm property. In our former papers[12, 13] we described some of the uses of such stronger orderings in term rewriting theory.

We first recall Ackermann's ordinal system from[4], which is considered to be the partial (quasi) ordered version of Ackermann[1] (also cf. Schütte[14]).

> **Definition.** $<A_n(F,C),\succ>$ (The Ackermann ordering on $A_n(F,C)$ - note: we may consider these terms as labeled trees with internal nodes taken from the set F, and leaves taken from the set C). Let F and C be well-founded by $>$.
>
> **Case 1.** if $s,t \in C$, then $s\ t$ if $s > t$ in C. If $s \in C$ and $t \notin C$, then $t \succ s$ holds but $s \succ t$ does not.
>
> **Case 2.** Let $s \equiv f(s_1, \ldots, s_n)$ and $t \equiv g(t_1, \ldots, t_n)$. Then $s \succ t$ holds if
>
> (1) $s_i \succeq t$ for some i $(1 \leq i \leq n)$,
>
> (2) $f > g$, $s \succ t_1, \ldots, s \succ t_n$,
>
> (3) $f = g$, $s_1 \approx t_1, \ldots, s_{i-1} \approx t_{i-1}, s_i \succ t_i, s \succ t_{i+1}, \ldots, s \succ t_n$ for some i $(1 \leq i \leq n)$, where $\approx$ means the permutative congruence.
>
> **Case 3.** let $s \equiv s_1 \# \cdots \# s_m$ and $t \equiv t_1 \# \cdots \# t_l$. Then $s \succ t$ iff $[s_1, \ldots, s_m] \succ\succ [t_1, \ldots, t_l]$, where $\succ\succ$ is the multiset ordering induced by $\succ$.

We now give our main theorem.

Theorem. (1) Any simplification ordering on unstructured trees (with finite signatures) is an initial segment of the subsystem $A_1(\omega,\omega)$ of Ackermann's system of ordinals.

(2) Any simplification ordering on general trees (combined of structured and unstructured branches) is an initial segment of Ackermann's system of ordinals.

Sketch of Proof. Without loss of generality we consider an extension of a given simplification ordering to a total ordering. It is well-known that the well-foundedness of a simplification ordering can be shown as a direct corollary to Kruskal's tree embedding theorem. On the other hand, the proof by the so called minimal bad sequence of Nash-Williams[10] can be naturally formalized in a subsystem of the system ID_1 of non-iterated inductive definitions. In fact, the Kruskal theorem for unstructured labeled trees is formulated in the system G of non-iterated inductive definitions with non-nested implications (see 3. of Okada[11] for the precise definition of G). The Kruskal theorem for general labeled trees is formulated in the system G_1, which is obtained from G by allowing one-nested implications. Therefore, the proof of well-foundedness of these two kinds of simplification orderings are naturally formulated in G and G_1, respectively (see 3. of Okada[11] for an example of formalizing a well-foundedness proof of a simplification ordering for unstructured trees in G). On the other hand, a traditional proof theoretic reduction argument shows that the consistency of G and G_1 can be proved by the transfinite induction on the ordinal structures $\phi_\omega 0$ (which has the same size as $A_1(\omega,\omega)$), and $\lim_n |A_n(\omega,\omega)|$, respectively. Hence by Gödel's second incompleteness theorem, they cannot be shown to be well-founded in G and G_1, respectively. In other words, those two ordinals are the critical ordinals for G and G_1, respectively. This concludes the statement of the theorem.

If the reduction ordering is not compatible with the subterm property then we cannot use a simplification ordering to prove termination, and must instead use a larger ordering. Examples of the use of such orderings to prove termination of rewriting systems on general trees are given in the paper of the first author[12].

4. COMPUTATIONAL POWER OF A CONVERGENT SYSTEM BASED UPON THE COMPLETION ORDERING

In this section we will show how the size of the ordinal structure used to orient the rewrite rules generated by the completion process determines the upper bound of the computational power of the completed system.

Definition (ordinal recursive functions). Let α be an ordinal number, and $<$ a primitive recursive well-ordering on the natural numbers of order type α (we are assuming the structure $<\alpha,<>$ is coded in the natural numbers). The class of α-recursive functions is defined by the following schemata:

(i) $S(a)=a+1$ (Successor)
(ii) $Z(a_1, \ldots, a_n)=0$ (Zero)
(iii) $P_{j_n}(a_1, \ldots, a_n)=a_j (1 \leq j \leq n)$ (Projection)
(iv) $f(a_1, \ldots, a_n)=g(h_1(a_1, \ldots, a_n), \ldots, h_m(a_1, \ldots, a_n))$,
where g and h_j are known to be α-recursive.
(v) (α-recursion)

$$f(a_1, \ldots, a_n) = \begin{cases} h(f(k(a_1,..,a_n),a_2, \ldots, a_n),a_1, \ldots, a_n), \\ \qquad \text{if } k(a_1,..,a_n) < a_1 \\ g(a_1, \ldots, a_n), \text{ otherwise} \end{cases}$$

where g, h and k are previously defined α-recursive functions.

The idea of (v) is that $f(a,a_2, \ldots, a_n)$ is either defined outright, or in terms of $f(b,a_2,..,a_n)$ for certain $b<a$.

If we take ω for α and the natural ordering of ω for $<$, then the class of α-recursive functions is exactly the same as the class of primitive recursive functions.

Definition (provably recursive functions). For a given mathematical system S, the class of provably recursive functions in S is the subclass of the (total) recursive for which termination of the calculation procedure for each input (i.e. each natural number) is uniformly proved in the system S. In other words, a recursive function $f(x)$ is provably recursive in S iff $f(x)=U(\mu y T(e,x,y))$ for some e in the sense of Kleene, and $S \vdash \forall x \exists y T(e,x,y)$, where U is a fixed primitive recursive function and T is a fixed primitive recursive predicate called Kleene's predicate[9].

Definition. A rewrite system R is computable by a recursive function f, if (under a suitable coding - Gödel numbering - on the natural numbers) we have a recursive function f such that for any ground term s there exists a unique normal form t of s (i.e. $s \xrightarrow{*}_R t$ and no rewrite rule in R can be applied to t) and $f(\hat{s})=\hat{t}$, where $\hat{s}$ and $\hat{t}$ are the Gödel numbers of s and t respectively.

The following is an alternate characterisation of α-recursive functions.

Proposition. For a convergent rewrite system, the following are equivalent:

(1) the system is computable by an α-recursive function.

(2) the rank function of the system is α-recursive, where the rank function f of a system R is defined as $f(|s|)=|t|$ for any term s and the normal form t of s, and $|u|$ is the length of term u.

(3) the growth of the number of steps in any reduction sequence in the system is bounded by an α-recursive function, more precisely, for a given term s, consider a function f such that $f(|s|)$ is number of steps in the shortest reduction sequence for s, then f is α-recursive.

Proposition. (1) For any finite sets F and C, any term rewriting system resulting from completion with any simplification ordering (on unstructured trees) is computable by a Γ_0-recursive function.

(2) For any finite sets F and C, any term rewriting system resulting from completion with any simplification ordering on binary structured trees is computable by an ϵ_0-recursive function.

(3) For any well-founded set C whose rank is less than ϵ_0, any term rewriting system resulting from completion with the ordering $A_1(\{f\},C)$ is computable by an ϵ_0-recursive function.

A well known result from proof theory is that the class of $<\epsilon_0$-recursive functions (i.e. all α-recursive functions for any $\alpha<\epsilon_0$) is exactly the class of provably recursive functions in the system PA (Peano Arithmetic), and the class of $<\Gamma_0$-recursive functions is exactly the class of provably recursive functions in Predicative Analysis (cf. Feferman[6]). On the other hand, Gentzen type reduction rules for proof trees used in in the consistency proof of Peano Arithmetic (Predicative Analysis[14]) can be interpreted as term rewriting systems where the terms are proof trees. It is known from proof theory that these rewriting systems are embeddable into an ordering of order type ϵ_0 (Γ_0, respectively), but not embeddable into any ordering of order type less than ϵ_0 (Γ_0, respectively).

Theorem. If F and C are finite (or if the well-foundedness of F and C is provable in ID_1), any rewrite system resulting from completion with any simplification ordering is computable by a provably recursive function of G_1, and hence ID_1.

With a given ordinal notation system up to the ordinal α satisfying the monotonicity property, and stable under substitutions, one can construct a rewriting system which calculates an exactly α-recursive function (i.e. α-recursive, but not β-recursive, for $\beta<\alpha$). Some examples of such functions may be found in the papers by Okada[12], and Dershowitz and Okada[5].

References

1. Ackermann, W., "Konstruktiver Aufbau eines Abschnitts der zweiten Cantorischen Zahlenklasse," *Math Z.*, vol. 53, pp. 403-413, 1951.
2. Bachmair, L., N. Dershowitz, and J. Hsiang, "Orderings for Equational Proofs," in *Proc. IEEE Symp. Logic in Computer Science*, pp. 346-357, Cambridge, Massachusetts, 1986.
3. Bachmair, L. and N. Dershowitz, "Critical Pair Criteria for Completion," *J. Symbolic Computation*, vol. 6, pp. 1-18, 1988.
4. Dershowitz, N. and M. Okada, "Proof-Theoretic Techniques for Term Rewriting Theory," in *Proc. of the Third Annual Symposium on Logic in Computer Science*, Edinburgh, 1988.
5. Dershowitz, N. and M. Okada, "Well-quasi Ordering on Trees in Term-rewriting Theory, Proof Theory, and Graph Theory," in preparation, 1989.
6. Feferman, S., "Systems of Predicative Analysis," *Journal of Symbolic Logic*, vol. 29, pp. 1-30, 1964.
7. Huet, G., "A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm," *Journal of Computer and System Sciences*, vol. 23, pp. 11-21, 1981.
8. Kapur, D., D. R. Musser, and P. Narendran, "Only Prime Superpositions Need to be Considered in the Knuth-Bendix Completion Procedure," *J. Symbolic Computation*, vol. 6, pp. 19-36, 1988.
9. Kleene, S., *Introduction to Metamathematics,* Van-Nostrand, 1958.
10. Nash-Williams, C. St. J. A., "On Well-quasi-ordering Finite Trees," *Proc. Cambridge Philos. Soc.*, vol. 59, pp. 833-835, 1963.
11. Okada, M., "A Theory of Weak Implications," *Journal of Symbolic Logic*, vol. 53, no. 1, 1988.
12. Okada, M., "Note on a Proof of the Extended Paris-Kirby Game for Labeled Finite Trees," *European Journal of Combinatorics*, 1988.
13. Okada, M., "A Logical Analysis on the Theory of Conditional Rewriting," in *Conditional Term Rewriting Systems*, ed. S. Kaplan, Springer-Verlag, Berlin, 1988.
14. Schütte, K., *Beweistheorie,* Springer, Berlin, 1960.
15. Steele, A., "Simplifying the Knuth-Bendix Completion Procedure for Equational Systems," Masters Thesis, Concordia University, 1989.

PARALLEL BINARY SEARCH

SELIM G. AKL and HENK MEIJER
Department of Computing and Information Science
Kingston, Ontario K7L 3N6
CANADA

ABSTRACT

Two arrays of numbers sorted in nondecreasing order are given: an array A of size n and an array B of size m, where $n < m$. It is required to determine, for every element of A, the smallest element of B (if one exists) that is larger than or equal to it. We show how to solve this problem on the EREW PRAM in $O(\frac{\log m \log n}{\log \log m})$ time using n processors. The solution is then extended to the case where fewer than n processors are available. This yields an EREW PRAM algorithm for the problem whose cost is $O(n \log m)$, which is $O(m)$ for $n \leq m / \log m$. Finally, we show how our solution leads to an improved parallel merging algorithm.

INTRODUCTION

Consider the following problem. Two arrays of numbers sorted in nondecreasing order are given: an array A of size n and an array B of size m, where $n < m$. It is required to determine, for every element of A, the smallest element of B (if one exists) that is larger than or equal to it. We call this the Multiple Search Problem (MSP). Sequentially, MSP can be easily solved in $O(m)$ time by a simple modification to the standard **merging** algorithm. A second approach would be to conduct a **binary search** on B for every element of A. This requires $O(n \log m)$ time, and is preferable to merging when $n < m / \log m$.

We are interested in parallel solutions to MSP. Our chosen model of parallel computation is the Parallel Random Access Machine (PRAM) with $p \geq 2$ processors [1]. In this model, the processors share a common memory which they use to read and write data and exchange intermediate results. Depending on whether or not two or more processors are allowed to gain access to the same memory location simultaneously, we can distinguish four variants of the PRAM:

1) The Exclusive-Read Exclusive-Write (EREW) PRAM, where no two processors are allowed to read from or write into the same memory location at the same time;

This research was supported by the Natural Sciences and Engineering Research Council of Canada under Grants #A-3336 and #A-0282.

2) The Concurrent-Read Exclusive-Write (CREW) PRAM, where simultaneous reading is allowed from the same memory location, but not simultaneous writing;

3) The Exclusive-Read Concurrent-Write (ERCW) PRAM, where simultaneous writing is allowed into the same memory location, but not simultaneous reading;

4) The Concurrent-Read Concurrent-Write (CRCW) PRAM, where both simultaneous reading from and writing into the same memory location are allowed.

Of the four models the EREW PRAM is the weakest in terms of the time required to solve a problem in parallel. If an algorithm for any of the other three models runs in time T with p processors, then it can be simulated on the EREW in time T log p [1]. Let the cost of a parallel algorithm be defined as the product of its running time and the number of processors it uses. This quantity essentially expresses the running time of the parallel algorithm when simulated on a single-processor computer. Consequently, when designing parallel algorithms, we strive to ensure that their cost does not exceed the best sequential solutions known.

Parallel solutions to MSP with different properties can be obtained using a variety of approaches. For example, if p processors are available, then an algorithm for the EREW PRAM whose running time is $O(\frac{m}{p}\log m + \log m)$ can be obtained based on **parallel sorting** [3]. When $p = m$, the algorithm runs in $O(\log m)$ time. The cost of this solution exceeds the sequential running time of the solution to MSP based on merging. By contrast, **parallel merging** can be used to obtain a solution whose cost matches sequential merging. The algorithm in [2] runs on the EREW PRAM, in $O(\frac{m}{p}+\log^2 m)$ time; when $p \leq m/\log^2 m$ the cost is $O(m)$. In [4] a CREW PRAM algorithm is given that runs in $O(\frac{m}{p} + \log\frac{m}{p} + \log\log p)$. This algorithms has a cost of $O(m)$ when $p \log\frac{m}{p} = O(m)$, but its cost is larger when simulated

on an EREW PRAM. A third solution uses the binary search approach referred to above for solving MSP. A straightforward implementation of this algorithm on the CREW PRAM runs in $O(\log m)$ using n processors. This solution, whose cost is $O(n \log m)$ as in the sequential case, is superior to the previous two when $n < m/\log m$ and **only n processors are available.** A direct simulation of this implementation on the EREW PRAM requires $O(\log m \log n)$ time.

In this paper, we show how MSP can be solved on the EREW PRAM in $O(\frac{\log m \log n}{\log\log m})$ time using n processors. The solution is then extended to the case where fewer than n processors are available. This yields an EREW PRAM algorithm for MSP whose cost is $O(n \log m)$, which is $O(m)$ for $n = O(m/\log m)$. Finally, we show how our solution to MSP in turn leads to an improved parallel merging algorithm for certain values of n, m and p.

PARALLEL BINARY SEARCH WITH n PROCESSORS

We begin by defining MSP formally. Let A[0..n-1] and B[0..m-1] be two arrays sorted in non-decreasing order. It is required to construct the array C[0..n-1] defined by:

$$C[i] = \min_{0 \leq j < m} \{ j \mid A[i] \leq B[j] \} \qquad \text{if } A[i] \leq B[m-1]$$

$$C[i] = m \qquad \text{otherwise.}$$

The following algorithm solves MSP in parallel on the EREW PRAM with n processors denoted $P_0, P_1, \ldots, P_{n-1}$.

```
procedure pbs (A[s..t], B[x..y])
    { This procedure solves MSP through parallel binary search }
    { Sequential steps are assumed to be executed by $P_s$ }
if $s \leq t$ then
        mid := $\lfloor (s+t)/2 \rfloor$
        if $A[mid] \leq B[x]$ then
                for i := s to mid do in parallel
                   C[i] := x { This step is executed by $P_i$ }
                end for
                pbs (A[mid+1..t], B[x..y])
        else
                if A[mid] > B[y] then
                        for i := mid to t do in parallel
                                C[i] := y + 1 { This step is executed by $P_i$ }
                        end for
                        pbs (A[s..mid-1], B[x..y])
                else
                        { C[mid] is determined using sequential binary search }
                        z := $\min_{0 \leq j < m} \{ j \mid A[mid] \leq B[j] \}$
                        C [mid] := z
                        do in parallel
                                pbs (A[s..mid-1], B[x..z-1]) and
                                pbs (A[mid+1..t], B[z..y])
                        end do
                end if
        end if
end if. □
```

If pbs (A,B) is called with an array A of length n, the depth of recursion is O(log n). Since for each call to pbs(A,B) a binary search of array B has to be executed, it follows that the algorithm runs in O(log n log m) time. That the algorithm is correct derives from the fact that if A[mid] > B[y] then C [i] is y + 1 for mid $\leq$ i $\leq$ t, even if y = m - 1.

A speedup can be obtained by dividing the array A into k+1 sections of approximately equal size. The k elements of A that separate these sections are located in array B in parallel. To prevent read conflicts the k searches start at different times. Thus, the searches for elements 1, 2, ..., k start at times 1, 2, ..., k, respectively. After these searches are completed, the elements in the k+1 sections can be located recursively. The algorithm is described below.

```
procedure faster-pbs (A[s..t], B[x..y])
    { This procedure solves MSP through multiple parallel binary searches }
if s ≤ t then
        { interval [s..t] is divided into k+1 sections; }
        { the k splitting elements are stored in array SP[1..k] }

        for i := 1 to k do in parallel
                { $P_{s+i-1}$ computes the index of the ith splitting element }
                SP [i] := s + $\lfloor \frac{t-s+1}{k+1} \cdot i \rfloor$
        end for

        { all k splitting elements are located in B }
        for i = 1 to k do in parallel
                $P_{SP[i]}$ starts at time i and finds C[SP[i]] using binary search
        end for

        { the k+1 subproblems are solved recursively }
        for i = 0 to k do in parallel
                if i=0 then left.A := s;   left.B := x
                else left.A := SP[i] + 1;   left.B := C[SP [i]]
                end if

                if i < k then right.A := SP[i+1]-1;   right.B := C[SP[i+1]]-1
                else right.A := t;   right.B := y
                end if
                if left.B > right.B { i.e. if the B subinterval is empty } then
                        for j = left.A to right.A do in parallel
                                $P_j$ assigns left.B to C[j]
                        end for
                else faster-pbs(A[left.A..right.A], B[left.B..right.B])
                end if
        end for
end if. □
```

Since the binary searches for the k splitting elements start at different times, no read conflicts occur. Therefore the k splitting elements can be located in $O(\log m + k)$ time. If we assume that in all recursive calls k remains constant, then the running time $T(n,m,k)$ of the algorithm satisfies:

$$\begin{aligned} T(n,m,k) &= O(\log m + k) + T(\frac{n}{k}, m, k) \\ &= O((\log m + k)(\frac{\log n}{\log k})). \end{aligned}$$

If at the start of the algorithm k is set equal to log m, then an asymptotic running time complexity of $O(\frac{\log m \log n}{\log \log m})$ is obtained, which is a factor of log log m faster than a direct EREW simulation of the $O(\log m)$ CREW algorithm with n processors. This result also implies that the n elements can be located in $O(\log m)$ time for $n = \log^{\alpha} m$, where α is an arbitrary constant.

SEARCHING WITH FEWER PROCESSORS

Suppose that the number of processors available on the EREW PRAM is p, where $p < n$. Procedure faster-pbs can be used until all p processors are occupied. Since the number of active processors increases by a factor of k at each level of recursion, there are $O(\frac{\log p}{\log k})$ iterations before all processors are active. The remaining p intervals of array A can be simultaneously located in the corresponding p intervals of array B by the p processors, whereby each interval is solved sequentially. Since the size of the remaining intervals of A is approximately n/p, this final step can be executed in $O(\frac{n}{p} \log m)$ time.

The total running time is therefore

$$O((\log m + k)(\frac{\log p}{\log k}) + \frac{n}{p} \log m),$$

with a cost of $O(\frac{p \log m \log p}{\log \log m} + n \log m)$ when $k = \log m$. This cost is $O(n \log m)$ for $p \leq \frac{n \log \log m}{\log n}$ and hence matches the running time of the corresponding sequential solution.

two sections lead to the following observation regarding adaptive algorithms for parallel merging.

Assume that two sorted arrays A of size n and B of size m, with $n < m$, are to be merged in parallel to form an array D of length $m+n$. The EREW PRAM algorithm of [2] performs the merge in $O(\frac{m}{p} + \log^2 m)$ time using p processors. Another way of merging A and B would be to first apply the algorithm of section 3 with $k = \log m$; it runs in $O(\frac{\log m \log p}{\log \log m} + \frac{n}{p} \log m)$ time, where $p < n$. This gives for each element A[i] its position in array D, given by $i + C[i]$. The n elements of A are thus deposited in D by the p processors in $O(\frac{n}{p})$ time, where $p < n$. We then determine for each element B[i] its position in array D as follows. We associate with B[i] a count E[i] initialized to 1, which can be done in $O(m/p)$ time. The array C is then used to update up to n positions of array E: if $C[i] = j$, then E[j] is incremented by 1. Since several elements of C may be equal and concurrent writes are disallowed, this operation requires $O(\frac{n}{p}\log p)$ time. The position of B[i] in array D is obtained by computing $E[0] + E[1] + \ldots + E[i] - 1$, which can be done in parallel for all B[i] in $O(\frac{m}{p}+\log p)$ time [6]. Finally the m elements of B are deposited in D by the p processors in $O(m/p)$ time. The entire algorithm thus requires

$$\frac{\log m \log p}{\log \log m} + \frac{n}{p} \log m + \frac{m}{p}$$

time. For $n < \frac{m}{\log m}$, the new merging algorithm is faster than that of [2]. It is also cost-optimal for a wider range of p, namely $p \leq \frac{m \log \log m}{\log^2 m}$.

In section 1 we observed that merging can be used to solve the problem of multiple search. The result of the previous paragraph suggests that an algorithm for multiple search in turn leads to an improved algorithm for parallel merging.

CONCLUSIONS

The table below gives a comparison of the new parallel binary search algorithm with some existing parallel solutions for the MSP problem.

algorithm	time	cost
(a) parallel sorting EREW [3]	$O(\frac{m}{p}\log m + \log m)$	$O(m \log m)$
(b) parallel merging EREW [2]	$O(\frac{m}{p}+\log^2 m)$	$O(m+p \log^2 m)$
(c) parallel merging CREW [4]	$O(\frac{m}{p} + \log \frac{m}{p} + \log \log p)$	$O(m + p \log \frac{m}{p} + p \log \log p)$
(d) parallel binary search CREW	$O(\frac{n}{p} \log m)$	$O(n \log m)$
(e) parallel binary search EREW	$O(\frac{\log m \log p}{\log \log m}+\frac{n}{p}\log m)$	$O(\frac{p \log m \log p}{\log \log m}+n \log m)$

A summary of the main results of this paper is given below

- for $n = \log^{\alpha} m$, where α is a constant, and $p=n$, algorithm (e) has the same asymptotic time complexity as algorithm (d).
- for $p \leq \min (\frac{n \log \log m}{\log n}, n)$ the cost of algorithm (e) is $O(n \log m)$, which is better than the cost of algorithms (a), (b), (c) and (d) for $n \leq m/\log m$.
- algorithm (e) with $p < n$ processors can be used to merge two arrays in

$$\frac{\log m \log p}{\log \log m} + \frac{n}{p}\log m + \frac{m}{p}$$

time. For $n \leq m/\log m$ we derive that the time complexity is less than or equal to

$$\frac{\log^2 m}{\log \log m} + \frac{m}{p}$$

and the cost is $O(m)$ for $p \leq \frac{m \log \log m}{\log^2 m}$.

An EREW PRAM algorithm for the Multiple Search Problem, different from the one described in this paper, is proposed in [7]. The algorithm uses p processors and runs in $O(\log m + \log^2 p + \frac{n}{p} \log m)$ time. When $p = O(\frac{n}{\log n})$ and $n \leq m^{\alpha}$, with $0 < \alpha < 1$, the cost of the algorithm matches the running time of the fastest known sequential algorithm for this problem, namely $O(n + n \log \frac{m}{n})$ [5].

REFERENCES

[1] Akl, S.G., **The Design and Analysis of Parallel Algorithms**, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[2] Akl, S.G., and Santoro, N., Optimal parallel merging and sorting without memory conflicts, **IEEE Transactions on Computers**, Vol. C-36, No. 11, November 1987, pp. 1367-1369.

[3] Cole, R., Parallel merge sort, **SIAM Journal on Computing**, Vol. 17, No. 4, August, 1988, pp. 770-785.

[4] Kruskal, C.P., Searching, merging, and sorting in parallel computation, **IEEE Transactions on Computers**, Vol. C-32, No. 10, October 1983, pp. 942-946.

[5] Lang, S.D., and Deo, N., Recursive batched binary searching of sequential files, University of Central Florida, to appear in **The Computer Journal**, 1990.

[6] Meijer, H., and Akl, S.G., Optimal computation of prefix sums on a binary tree of processors, **International Journal of Parallel Programming**, Vol. 16, No. 2, April 1987, pp. 127-136.

[7] Meijer, H., and Akl, S.G., Parallel binary search with delayed read conflicts, **Technical Report No. 89-257**, Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada, June 1989.

Parallel Testing for the Consecutive Ones Property and Transformable Convex Bipartite Graphs and Finding a Maximum Matching

LIN CHEN*
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210

YAACOV YESHA†
Department of Computer Science
University of Maryland Baltimore County
Baltimore, Maryland 21228

University of Maryland
Institute for Advanced Computer Studies
College Park, Maryland 20742

Abstract

In this paper, we give the first efficient parallel algorithms for three problems on CRCW PRAM. Specifically, we show

- the consecutive 1's property for rows can be recognized in $O(\log n + \log^2 m)$ time with $O(m^2 n + m^3)$ processors, where $m \times n$ is the size of the input matrix;
- a transformable convex bipartite (TCB) graph can be decided in $O(\log^2 n)$ time with $O(n^3)$ processors, where n is the number of the vertices of the input graph;
- a maximum matching in a TCB graph can be computed within the same resource bounds as above.

The algorithm for the consecutive 1's property gives a useful tool for solving many problems. The next algorithm lays the foundation for solving many other problems on TCB graphs. The algorithm for the maximum matching problem broadens the class of graph for which there are known fast parallel algorithms.

---

*Supported in part by the Office of Research and Graduate Studies of the Ohio State University.

†Supported in part by the National Science Foundation under grant DCR-8606366. Part of the work of this author was done at the Department of Computer and Information Science, The Ohio State University.

1 Introduction

Due to the dramatic drop in the cost of the hardware components, parallel computers are becoming more and more popular, which in turn, makes the study of parallel algorithms more and more important.

A problem is said to be in $(R)NC^k$ if there is a (randomized) algorithm which solves the problem in $O(log^k n)$ time using polynomially many processors. $(R)NC$ is defined as the union of $(R)NC^k$ for all integers k's. Such an algorithm is called an $(R)NC$ algorithm. Given a problem Q, the first important question is: is Q in NC? One of the well known open problems is: is the maximum matching problem in NC?

The study of the maximum matching problem plays an important part in the development of computer algorithms. It has led to conceptual breakthroughs in this area. In fact, the characterization of 'tractable problems' as 'polynomial time solvable problems' was first proposed by Edmonds [9] in the context of the general matching problem. Hopcroft and Karp [16] give the fastest sequential algorithm for bipartite graphs. Lipski and Preparata [19] give the fastest sequential algorithm for convex bipartite graphs. Galil [12] has provided a very good survey on the maximum matching problem.

Dekel and Sahni [8] gave an NC algorithm for finding a maximum matching in a convex bipartite graph. Later, Karp et al [17] showed that the maximum matching problem is in RNC. Recently, Kozen et al [18] and Helmbold and Mayr [15] found the maximum matching problem for co-comparability graphs is in NC. More recently, Dahlhaus and Karpinski [7] showed that the problem for strongly chordal graphs is in NC.

In this paper, we study another class of graphs called transformable convex bipartite graphs. This class is a natural and desirable generalization of the class of convex bipartite graphs. Unlike convex bipartite graphs, transformable convex bipartite graphs do not depend on the labeling or the order of the vertices. The class of transformable convex bipartite graphs is the closure of the class of convex bipartite graphs with respect to graph isomorphism. We present an efficient parallel algorithm which recognizes transformable convex bipartite graphs and transforms them into convex bipartite graphs. This also solves the problem of recognizing the consecutive retrieval property [10]. In fact, the transformation itself is a very interesting problem. It has applications in many fields such as genetics, archaeology, psychology, information retrieval, and so on. Readers are referred to [13] [20] for more details. The first sequential algorithm for the transformation is in Fulkerson and Gross [11]. And later, Eswaran [10] designed another sequential algorithm and developed necessary and sufficient conditions for the existence of the transformation. Booth and Leuker [1] gave an efficient sequential algorithm for recognizing the existence of the transformation in the context of testing for the consecutive 1's property. In this paper, we successfully parallelize a modified version of Fulkerson and Gross's algorithm. The result here makes much larger the class of graphs for which the maximum matching problem is known to be in NC. Notice that bipartite permutation graphs are a subclass of transformable convex bipartite graphs. So we have also given a fast parallel algorithm for finding a maximum matching in a bipartite permutation graph. Building on the work here, we have given NC algorithms which test for interval graphs and the circular 1's property [5], and NC algorithms which test for bipartite permutation graphs and isomorphism of bipartite permutation graphs, and which find a Hamiltonian path and cycle and solve a variant of the crossing number problem for a bipartite permutation graph [4], and NC algorithms which recognize proper interval graphs and proper circular-arc graphs and which solve various problems on these graphs [2] [3]. We can further give an efficient algorithm to test the isomorphism of transformable convex bipartite graphs.

2 Preliminaries

A convex bipartite graph $G = (X, Y, E)$ is a bipartite graph which satisfies the following conditions:
a. X and Y are ordered sets;
b. If $(x_i, y_j) \in E$ and $(x_i, y_{j+k}) \in E$, then $(x_i, y_{j+q}) \in E$, for $1 \leq q < k$.

A bipartite graph $G = (X, Y, E)$ is called a transformable convex bipartite graph if it is isomorphic to a convex bipartite graph.

Let $G = (X, Y, E)$ be a bipartite graph. A $(0,1)$-matrix B is called the bipartite graph matrix of G if B_{ij} is 1 if and only if $(x_i, y_j) \in E$.

We say that a $(0,1)$-matrix M has the consecutive 1's property for rows if there exists a permutation matrix P such that the 1's in each row of MP occur consecutively.

For a $(0,1)$-matrix B, it is convenient, and imposes no loss of generality in studying the consecutive 1's property, to assume that no row vector and no column vector are all 0's, and that no two row vectors are the same. In the sequel, we will make these assumptions unless otherwise stated and refer to such a matrix B as proper.

Vector a and vector b over $\{\ 0,\ 1\ \}$ are said to intersect if their inner product $a \cdot b > 0$. They overlap if they intersect and $a \cdot b < min(a \cdot a, b \cdot b)$. If $a \cdot b = b \cdot b$, we say that vector a contains vector b.

An undirected graph $S(B) = (V, E)$ is called an intersection graph of a $(0,1)$-matrix B if there exists a bijection f: $V \longrightarrow$ the set of row vectors of B such that for any two distinct vertices v_i and v_j in V, $\{v_i, v_j\} \in E$ if and only if $f(v_i)$ and $f(v_j)$ intersect. Similarly, an undirected graph $L(B) = (V, E)$ is called an overlap graph of a $(0,1)$-matrix B if there exists a bijection f: $V \longrightarrow$ the set of row vectors of B such that for any two vertices v_i and v_j in V, $\{v_i, v_j\} \in E$ if and only if $f(v_i)$ and $f(v_j)$ overlap.

Hereafter, we will use the vertex of an overlap graph to indicate a row vector of the corresponding $(0,1)$-matrix, and vice versa, if no confusion can arise.

A directed graph $T(B) = (V, E)$ is called the component graph of a $(0,1)$-matrix B, if there exists a bijection f: $V \longrightarrow$ the set of the components of the overlap graph such that for any two distinct vertices v_i and v_j in V, $(v_i, v_j) \in E$ if and only if a row vector x_p in the component corresponding to v_i contains a row vector x_q in the component corresponding to v_j.

A directed graph $C(B) = (V, E)$ is called a containment graph of a $(0,1)$-matrix B if there exists a bijection $f : V \longrightarrow$ the set of row vectors of B such that for any two distinct vertices v_i and v_j in

V, $(v_i, v_j) \in E$ if and only if $f(v_i)$ contains $f(v_j)$. A containment (or component) graph is said to be reduced if we omit the arcs that are implied by transitivity.

The parallel computation model used in this paper is a parallel random access machine (PRAM for short) Here we allow concurrent read and concurrent write(CRCW). In case of concurrent write, we only allow several processors to write the same value into the same memory cell.

Before we prove the theorem which is important to our algorithm design, we first prove a lemma.

Lemma 1 *If an undirected graph G is connected and contains at least one vertex, then there exists a vertex v such that $G - \{v\}$ is also connected.*

Proof. Let T be a spanning tree of G, and v be a leaf of T. Then $T - \{v\}$ is a spanning tree of $G - \{v\}$. Therefore, $G - \{v\}$ is connected. □

Theorem 1 *If a $(0,1)$-matrix has the consecutive 1's property for rows, and if the overlap graph is connected, then the matrix which can be obtained by permuting the columns and which has consecutive 1's in each of its rows is unique up to rotation.*

Proof. Apply induction on the number of rows of the matrix. If $n = 1$ or $n = 2$, it is trivially true. Recall the assumption that no row and no column are all 0's. Suppose it is true for $n = k (k > 1)$. Consider the case $n = k + 1$. Let the overlap graph be denoted by G. Permute the columns of the matrix so that the resulting matrix, denoted by M, has consecutive 1's in each of its rows. Since G is connected, by the above lemma, there exists at least one vertex v such that $G - \{v\}$ is also connected. Delete the corresponding row from the matrix M. The resulting matrix, M', also has consecutive 1's in each of its rows. By induction hypothesis, M' is unique up to rotation. Now look at M. Since G is connected, there exist v', v'' such that $\{v', v\}$ and $\{v', v''\}$ are edges of G. Observe that the inner products of $v \cdot v'$, $v' \cdot v''$ and $v'' \cdot v$ are not affected by the permutation of columns. The consecutive 1's for v' and v'' in M determines the unique location for the sequence of 1's corresponding to v. This completes the proof. □

3 The Algorithms

We now give the algorithm ConsecRow(M, M', *ok*) which decides whether the (0, 1)-matrix M has the consecutive 1's property for rows. If so, the procedure transforms M into a matrix M' which has consecutive 1's in each of its rows, and sets *ok* to *true*; otherwise, it returns with *ok* = *false*.

```
Procedure ConsecRow($M$, $M'$, ok)
    1. Construct the overlap graph and containment graph from the $(0,1)$-matrix M;
    2. Get the reduced containment graph;
    3. Get the connected components of the overlap graph ;
    4. Get the reduced component graph from the results of steps 2 and 3;
       comment: the resulting graph is a forest.
```

5. For every connected component of the overlap graph, get a spanning tree T ;

6. Get the component trees from the reduced component graph;

7. For every tree T, call Consec(T, MX, yes);
   comment: If the submatrix corresponding to T does not have the consecutive 1's property for rows, Consec(T, MX, yes) will return with yes = false; otherwise yes = true and MX is a matrix obtained from the matrix corresponding to T by permuting its columns so that the 1's in each row of MX appear consecutively.

8. If all the connected components of the overlap graph have the consecutive 1's property, then proceed with $ok = true$; otherwise quit with $ok = false$;
   comment: the matrix has consecutive 1's property if and only if all the components have consecutive 1's property.

9. For every component tree Tr, call Arrange(Tr, Matrix);
   Arrange(Tr, Matrix) will return in Matrix a convex bipartite graph corresponding to Tr.

10. Combine the matrix for every component tree into one matrix M'

End ConsecRow

We now analyze the time and the number of processors needed by procedure ConsecRow, and go into more details whenever appropriate. Assume the input matrix M is of size $m \times n$.

Step 1 can be done in $O(1)$ time as long as there are enough processors. For example, it can be done in the following way:

1. for $0 < i, j \leq m$, $l_{ij} := 0; c_{ij} := 0; d_{ij} := 0;$

2. for $0 < p, q \leq m, 0 < j \leq n$, if $m_{pj} = 1$ and $m_{qj} = 0$, then $d_{pq} := 1$;

3. for $0 < p, q \leq m$ and $p < q$

   if $d_{pq} = 1$ and $d_{qp} = 1$ then $\{ l_{pq} := 1; l_{qp} := 1 \}$;

   if $d_{pq} = 1$ and $d_{qp} = 0$ then $c_{pq} := 1$;

   if $d_{pq} = 0$ and $d_{qp} = 1$ then $c_{qp} := 1$

The resultant overlap graph is $(l_{ij})_{m \times m}$, and the containment graph $(c_{ij})_{m \times m}$. Substep 1 needs constant time and m^2 processors. Substep 2 can be done in $O(1)$ time with $O(m^2 n)$ processors. The last substep requires $O(1)$ time and $O(m^2)$ processors. Therefore, step 1 can be done in $O(1)$ time with $O(m^2 n)$ processors.

Step 2 can be done in $O(1)$ time with $O(m^3)$ processors. A possible way to do this is as follows:

1. for $0 < i, j \leq m, r_{ij} := c_{ij}$;

2. for $0 < i, j, k \leq m$, if $c_{ij} = 1$ and $c_{jk} = 1$ then $r_{ik} := 0$

The reduced containment graph is $(r_{ij})_{m \times m}$. The second substep is the dominating step. It can be done in $O(1)$ time with $O(m^3)$ processors.

The next step, getting connected components, can be done using the existing algorithm. The algorithm requires $O(log^2 m)$ time and $O(m^2/log^2 m)$ processors [6] .

Step 4 is to get the reduced component graph. We want to transform all the vertices of a reduced containment graph in the same connected component of the overlap graph into a single vertex of a reduced component graph. This only needs $O(1)$ time and $O(m)$ processors.

Step 5 can also be done using the existing algorithm [6] . Step 6 is to get the component trees. All we need to do is to get the connected components of the reduced component graph. So we can do this step in the same way as in the third step.

Step 7 is not as easy as the previous steps. The procedure can be described as follows:

```
Procedure Consec(T, MX, yes)
 Compute the size of the intersection of every two row vectors;
 Re-con(T, MX);
 If size of every pairwise intersection unchanged
      then yes := true
      else yes := false
End Consec
Procedure Re-con(T, MX)
If $|T| < c$ then obtain MX directly
 else Begin
      Get a balanced decomposition $(V_1, V_2)$ of $T$;
      Re-con($V_1, MX_1$);
      Re-con($V_2, MX_2$);
      Merge $MX_1, MX_2$ into $MX$
     End
End Re-con
```

The first step of procedure Consec is to compute the size of the intersection of every two row vectors. That is, for any two row vectors r_i and r_j, compute $r_i \cdot r_j$. In case $i = j$, $r_i \cdot r_j$ is the number of 1's in the row vector. Since the dimension of the row vector is n, we can get the size of the intersections of the row vectors in $O(\log n)$ time. Note that there are $O(m^2)$ pairwise intersections. So we need $O(m^2 n)$ processors.

Now we take a close look at procedure Re-con. In case $|T| < c$, we can get MX row by row. Each row can be regarded as an interval. For any three intervals a, b and c, if a and b overlap and b and c overlap, then we can decide whether a and c should be on the same side of b or not. If

$a \cdot c < min\{a \cdot b, b \cdot c\}$, then they should be put at the different sides of b. Otherwise, they should be put at the same side of b. Suppose we want to get a row corresponding to c, and we have known that c should be put at the right-hand side of b. Using a technique as in Shiloach and Vishkin [21], we can get the last point p of interval b in constant step. Since we know the length l of interval c, i.e., $c \cdot c$, and the size s of the intersection between b and c, i.e., $b \cdot c$, we know immediately that interval c is from point $p - s + 1$ to point $p - s + l$. We can handle the case $|T| < c$ in $O(1)$ time with $O(n)$ processors.

Next consider merging MX_1, and MX_2 into MX. Note that MX_1 and MX_2 are both interval representation matrices. By Theorem 1, we know that the interval representation is unique. Notice that two matrices have one interval in common. We use b to denote that interval. In MX_1, there exists an interval a such that a and b overlap. Likewise, there exists an interval c in MX_2 such that b and c overlap. By looking at these three intervals and comparing the values of $a \cdot b$, $b \cdot c$ and $a \cdot c$, we can decide whether a and c are at the right side of b. If not, on one of the two matrices, call it X, we need to perform the transformation which interchanges the column vectors X_i and X_{n+1-i} of X for $1 \leq i \leq n/2$, where n is the number of columns of X. After that, we can do the actual merging easily. This merging step can be done in constant time with $O(mn)$ processors. Notice that the resulting matrix is guaranteed to be an interval representation matrix, though the original matrix may not have the consecutive 1's property. The last step of procedure Consec will decide whether the resulting matrix can be obtained by transforming the original graph.

It is already known that a balanced decomposition can be found in $O(\log m)$ time [14]. Since the number of recursions in the procedure Re-con is bounded by $O(\log m)$, procedure Re-con can be done in $O(log^2 m)$ time. The processor requirement is $O(mn)$.

The last step of procedure Consec is to do some checking. The amount of resources needed is the same as in the first step of the procedure Consec. Therefore, the total time needed by step 7 is $O(\log n + log^2 m)$, and the total processor requirement is $O(m^2 n)$.

Step 8 is trivial. Its resource requirements are dominated by the other steps. For step 9, we adopt divide-and-conquer strategy and use a tree decomposition technique as in He and Yesha [14]. This step can be done in $O(log^2 m)$ with $O(mn)$ processors. For the last step, it is not difficult to see that the time needed is $O(\log m)$, and the number of processors needed is $O(mn)$.

From the above analysis, we can conclude that the overall time complexity for procedure ConsecRow is $O(\log n + log^2 m)$. The total number of processors needed is $O(m^2 n + m^3)$. Notice that we assume the input to procedure ConsecRow is a proper matrix. In case some rows only consist of 0's, we can simply discard those rows. If several rows are exactly the same, we can save one of them and delete the others. After the transformation, we duplicate some rows. All these can be done in constant time with no more processors. Hence we have the following theorem.

Theorem 2 *Testing if a $(0,1)$-matrix has the consecutive 1's property for rows, and if so, transforming the matrix into one with consecutive 1's in each of its rows can be done in $O(\log n + \log^2 m)$ time with $O(m^2 n + m^3)$ processors, where $m \times n$ is the size of the matrix.*

Based on *ConsecRow*, we can design an NC algorithm which recognizes transformable convex bipartite graphs. Notice that a bipartite graph is a transformable convex bipartite graph if and only if the bipartite graph matrix corresponding to each connected component has the consecutive 1's property for rows or columns.

```
Procedure Recog(B, yes)
    1. Obtain the connected components of $B$;
    2. For each connected component $C$ do in parallel
        begin
            ConsecRow($C$, $C'$, ok);
            D := $C^T$;
            ConsecRow($D$, $D'$, $ok'$);
            If $ok = false \wedge ok' = false$ then set yes to $false$ and return
        end;
    3. set yes to $true$
End Recog
```

Let us consider the resource complexity. Suppose the input graph has k connected component, where $k > 0$. There are n_i vertices in each connected component i. So $\sum_{i=1}^{k} = n$, where n is the number of vertices of the input graph. For each connected component i, the time needed is $O(\log^2 n_i)$ and processor requirement is $O(n_i^3)$. So the overall time complexity is $O(\max_{i=1}^{k} \log^2 n_i)$, which is $O(\log^2 n)$. The total processor requirement is:

$$\begin{aligned} O(n_1^3) + O(n_2^3) + \cdots + O(n_k^3) &\leq cn_1^3 + cn_2^3 + \cdots + cn_k^3 \\ &\leq c(n_1 + n_2 + \cdots + n_k)^3 \\ &= O(n^3) \end{aligned}$$

This result can be summarized as the following theorem.

Theorem 3 *A TCB graph can be recognized in $O(\log^2 n)$ time with $O(n^3)$ processors.*

Now let us consider the problem of finding a maximum matching of a transformable convex bipartite graph. For the sake of simplicity, we will assume the input graph is connected. If the graph is not connected, the solution is simply the combination of solutions of all the connected components.

We may obtain a maximum matching of a transformable convex bipartite graph using Dekel and Sahni's algorithm for finding a maximum matching in a convex bipartite graph. Before invoking their procedure, we need first transform the input graph into a convex bipartite graph. But even if we know the maximum matching of the convex bipartite graph, we may not know the maximum matching of the input graph. To tackle this difficulty, we will establish the correspondence between the edges of the graph before transformation and those after transformation. Suppose a bipartite graph matrix has the consecutive 1's property for rows. We permute the columns and get a convex bipartite graph. Then we try to obtain the correspondence by comparing the columns of the new and old matrix since every column of the new matrix is also a column of the old matrix and vice versa. But sometimes a matrix may have several identical rows, so before the transformation we add some rows to make every column distinct. Each of these rows appended consists of one 1 and several 0's. It is not difficult to see that this operation does not affect the consecutive 1's property for rows. In this way, we are able to establish the mapping of the edges. After we get the mapping, we delete the appended rows and invoke Dekel and Sahni's procedure. All these can be done in $O(log^2 n)$ time with $O(n^3)$ processors. So we have obtained the following result.

Theorem 4 *A maximum matching of a TCB graph can be computed in* $O(\log^2 n)$ *time with* $O(n^3)$ *processors.*

Acknowledgments

We are grateful to many people who helped us during this research. In particular, we would like to express our appreciation to Prof. C.L. Liu of the University of Illinois for his constructive suggestions, to Prof. Xin He of the State University of New York at Buffalo for helpful discussions, and to Prof. Vijay Vazirani of Cornell University for providing us with useful material, and last but not least, to Prof. M.T. Liu of the Ohio State University for his cheerful encouragement.

References

[1] K. S. Booth and G. S. Leuker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[2] L. Chen. Efficient parallel algorithms for several intersection graphs. In *Proc. 22nd Int'l Symp. on Circuits and Systems.* IEEE, 1989.

[3] L. Chen. NC algorithms for circular-arc graphs. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Proc. Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science, Vol. 382*, pages 291–302. Springer-Verlag, 1989.

[4] L. Chen and Y. Yesha. Fast parallel algorithms for bipartite permutation graphs. In *Proc. 26th Ann. Allerton Conf. on Communication, Control, and Computing*, pages 264–273. University of Illinois at Urbana-Champaign, 1988.

[5] L. Chen and Y. Yesha. Parallel algorithms: Testing for interval graphs and the circular ones property. Technical Report OSU-CISRC-9/88-TR29, The Ohio State University, 1988.

[6] F. Y. Chin, J. Lam, and I. Chen. Efficient parallel algorithms for some graph problems. *CACM*, 25:659–665, 1982.

[7] E. Dahlaus and M. Karpinski. The matching problem for strongly chordal graphs is in NC. Technical Report TR855-CS, Institut für Informatik, Universität Bonn, December 1986.

[8] E. Dekel and S. Sahni. A parallel matching algorithm for convex bipartite graphs. In *Proc. International Conference on Parallel Processing*, pages 178–184, 1982.

[9] J. Edmonds. Paths, trees and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[10] K. P. Eswaran. Faithful representation of a family of sets by a set of intervals. *SIAM J. Comput.*, 4:56–68, 1975.

[11] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

[12] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *Computing Surveys*, 18(1):23–38, March 1986.

[13] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Computer Science and Applied Mathematics. Academic Press, New York, 1980.

[14] X. He and Y. Yesha. Parallel recognition and decomposition of two terminal series parallel graphs. *Information and Computation*, 75:15–38, 1987.

[15] D. Helmbold and E. Mayr. Perfect graphs and parallel algorithms. In *Proc. International Conference on Parallel Processing*, pages 853–860, 1986.

[16] J. E. Hopcropt and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2:225–231, 1973.

[17] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in random NC. In *Proc. 17th Ann. ACM Symp. on Theory of Computing*, pages 22–32. Association for Computing Machinery, 1985.

[18] D. Kozen, U. V. Vazirani, and V. V. Vazirani. NC algorithms for comparability graphs, interval graphs, and unique perfect matchings. Technical Report TR86-799, Cornell University, December 1986.

[19] W. Lipski, Jr. and F. P. Preparata. Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. *Acta Informatica*, 15(4):329–346, August 1981.

[20] F. S. Roberts. *Discrete Mathematical Models with Applications to Social, Biological, and Environmental Problems.* Prentice-Hall, Englewood Cliffs, N.J., 1976.

[21] Y. Shiloach and U. Vishkin. Finding the maximum, merging, and sorting in a parallel computation model. *J. Algorithms*, 2:88–102, 1981.

OPTIMAL PARALLEL ALGORITHMS ON SORTED INTERVALS

SUNG KWON KIM[1]
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195

Abstract: We present $O(\log n)$ time, $O(n/\log n)$ processor EREW PRAM algorithms for finding a depth first search tree, a breadth first search tree, a maximum independent set, a minimum clique cover, a minimum dominating set, and a minimum coloring of interval graphs when the endpoints of the intervals are initially sorted by x-coordinates.

1 Introduction

Let A be a family of non-empty sets. A graph $G = (V, E)$ is called an intersection graph with a model A if there is an one-to-one correspondence between A and V and two vertices are adjacent if their corresponding sets have non-empty intersection. If A is a set of intervals in the real line, then G is an interval graph with an interval model A.

In his book [8], Golumbic said that "interval graphs are among the most useful mathematical structures for modeling real world problems", and gave many applications of interval graphs in archaeology, biology, job scheduling and storage information retrieval. Since an interval model is often assumed in the statements of many problems, it is natural to explore the complexity results on interval models of interval graphs.

Many sequential algorithms have been proposed for solving combinatorial optimization problems on interval graphs provided that an interval model is available [4, 8, 9, 10, 11, 14]. Bertossi and Bonuccelli [5] developed parallel algorithms for finding a maximum independent set, a minimum clique cover, and a minimum dominating set of an interval graph, all of which ran in $O(\log n)$ time using $O(n^2/\log n)$ processors in the CREW PRAM. Moitra and Johnson [15] improved the results of Bertossi and Bonuccelli by reducing the processor complexity to $O(n)$. They also presented an $O(\log n)$ time, $O(n^2/\log n)$ processor algorithm for computing a depth first search tree with a fixed starting vertex. Savage and Wloka [16] developed a CREW PRAM minimum coloring algorithm with running time $O(\log^2 n)$ using $O(n)$ processors and then improved it by reducing time complexity to $O(\log n)$ still using $O(n)$ processors in the EREW PRAM.

We present optimal $O(\log n)$ time, $O(n/\log n)$ processor EREW PRAM algorithms for finding a depth first search tree with an arbitrary starting vertex, a breadth first search tree with an arbitrary starting vertex, a maximum independent set, a minimum clique cover, a minimum dominating set, and a minimum coloring if the endpoints of the intervals are initially sorted by x-coordinates. Our results are new except the coloring. Note that the maximum clique problem can be easily solved by the parallel prefix computation.

We also present two problems and their parallel solutions which are of interest in their own right. One is called the "all tallest neighbors problem". Given n vertical line segments, sorted by x-coordinates, with bottom endpoints on the x-axis, the problem is to find for each segment the first vertical line segment hit by the right-going ray emanating from its top endpoint. Our solution takes $O(\log n)$ time and $O(n/\log n)$ processors in the EREW PRAM. The other is called the "red-blue matching problem". Given n red and blue points in the real line, we want to find a maximum matching between the red and blue points, where a red point can be matched to a blue point if the red point

---

[1] Research supported by NSF Grant CCR-8703196

is to the left of the blue point. We solve the matching problem in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM if the points are initially sorted by x-coordinates.

The next section gives parallel solutions to the all tallest neighbors and red-blue matching problems and discusses an application of the Euler tour technique. In section 3 we define some useful functions on intervals and show how to compute them efficiently. In the last section we give our algorithmic results on intervals.

2 Preliminaries

This section formally defines the all tallest neighbors and red-blue matching problems and presents their optimal parallel algorithms. The Euler tour technique is reviewed and an application is given.

We assume the following result from Ladner and Fischer [13] and Kruskal, Rudolph and Snir [12], stated as a lemma.

Lemma 2.1 [Parallel Prefix Computation] Given an array of integers $x_1, x_2, \cdots, x_n$, the prefix sums $s_k = \sum_{j=1}^{k} x_j$ for $1 \leq k \leq n$ can be computed in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

2.1 All Tallest Neighbors Problem

A vertical line segment in the plane can be represented by a triple (x, y, y') where x is the x-coordinate and y (resp. y') is the y-coordinate of the top (resp. bottom) endpoint. Since we shall deal with only *rooted* vertical line segments, i.e. those with $y > 0$ and $y' = 0$, a vertical line segment will be represented by an ordered pair (x, y), which is the coordinate of the top endpoint of the segment.

Suppose that we have a set of n (rooted) vertical line segments $S = \{s_i = (x_i, y_i) \,|\, 1 \leq i \leq n\}$. Assume that $x_i < x_{i+1}$ for $1 \leq i \leq n-1$. Two segments s_i and s_j for $i < j$ are said to be neighboring if we can draw a horizontal line intersecting them without intersecting any other segment s_k for $i < k < j$. In this case s_i is a left neighbor of s_j and s_j is a right neighbor of s_i. A segment s_i is said to be taller than s_j for $i \neq j$ if $y_i > y_j$. A segment s_j is the right tallest neighbor of s_i for $i < j$, denoted $RTN_S(s_i) = s_j$, if the horizontal line passing through the top endpoint of s_i intersects s_j without intersecting s_k for $i < k < j$. $RTN_S(s_i) = 0$ if no right neighbor satisfies the condition. The left tallest neighbor $LTN_S(\cdot)$ is analogously defined.

We shall show that if the line segments in S are sorted by x_i's, the right tallest neighbors can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM. An $O(\log n)$ time, $O(n)$ processor algorithm will first be given and then the processor complexity will be reduced to $O(n/\log n)$. Before describing the parallel algorithms, we give a linear time sequential algorithm. Assume that S is stored in a doubly-linked linear list in increasing order of x_i. Let s_{n+1} be a vertical line segment with x- and y-coordinates $= \infty$. Set $p \leftarrow s_n$ and $q \leftarrow s_{n+1}$. If q is taller than p, then $RTN(p) \leftarrow q$, $q \leftarrow p$ and $p \leftarrow PRED(p)$, else delete q from S and $q \leftarrow SUCC(q)$. Repeat the step until $p =$ NULL. $PRED(p)$ is the predecessor of p and $SUCC(p)$ is the successor of p in S. It is easy to verify that the algorithm works correctly and takes only $O(n)$ time.

2.1.1 Algorithm A: O(log n) time and O(n) processors

The $\sqrt{n}$-divide-and-conquer approach is used (e.g. see [1]). We begin by partitioning S into $\sqrt{n}$ subsets $T_1, T_2, \cdots, T_{\sqrt{n}}$, each of size $\sqrt{n}$, in such a way that the segments in T_i are to the left of those in T_{i+1}. Recursively solve the problem for each T_i in parallel. After the recursive calls, each segment $s \in T_i$ has $RTN_{T_i}(s)$ and $LTN_{T_i}(s)$, which are the right tallest neighbor and the left tallest neighbor of s in T_i, resp. If $RTN_{T_i}(s) \neq 0$ then it is the final solution, i.e. the right tallest neighbor of s in S.

Similarly if $LTN_{T_i}(s) \neq 0$ then it is the final solution.

Let h_i be the tallest line segment in T_i for $1 \leq i \leq \sqrt{n}$. Let $H = \{h_i \mid 1 \leq i \leq \sqrt{n}\}$. Compute $RTN_H(h_i)$ and $LTN_T(h_i)$ for $h_i \in H$. In order to do this, make $\sqrt{n}$ copies of H, assign $\sqrt{n}$ processors to each copy of H, and find $RTN_H(h_i)$ and $LTN_H(h_i)$ in the ith copy. All can be done in $O(\log n)$ time without concurrent reads.

All segments $s \in T_i$ to the left of h_i have $RTN_{T_i}(s) \neq 0$ and all segments $s \in T_i$ to the right of h_i have $LTN_{T_i}(s) \neq 0$. Let L_i (resp. R_i) be the set of segments s in T_i with $LTN_{T_i}(s) = 0$ (resp. $RTN_{T_i}(s) = 0$). For each $h \in H$, if $RTN_H(h) = h_i$, then h is assigned a copy of L_i. To do this, let $f_i = |\{h \in H \mid RTN_H(h) = h_i\}|$ for $1 \leq i \leq \sqrt{n}$. For each $1 \leq i \leq \sqrt{n}$, assign $f_i \cdot \sqrt{n}$ processors to L_i and make f_i copies of L_i, which can be done in $O(\log f_i)$ time.

Let $H_i = \{h \in H \mid LTN_H(h) = h_i\} \cup \{RTN_H(h_i)\} - \{0\} = \{h_{\alpha_i(1)}, h_{\alpha_i(2)}, \cdots, h_{\alpha_i(|H_i|)}\}$ with $\alpha_i(j) < \alpha_i(j+1)$. Let $\lambda_{i,1} = L_{\alpha_i(1)}$ and $\lambda_{i,k} = \{s \in L_{\alpha_i(k)} \mid s \text{ is taller than } h_{\alpha_i(k-1)}\}$ for $2 \leq k \leq |H_i|$. Let $\Lambda_i = \bigcup_{k=1}^{|H_i|} \lambda_{i,k}$. Note that $\sum_{i=1}^{\sqrt{n}} |H_i| \leq 2\sqrt{n}$ and $\sum_{i=1}^{\sqrt{n}} |\Lambda_i| \leq 2n$. Assign $|\Lambda_i| = |H_i| \cdot \sqrt{n}$ processors to each pair of R_i and Λ_i and merge them by y-coordinates. Observe that the vertical line segments in R_i are decreasing and those in Λ_i are increasing. Since $|R_i| \leq \sqrt{n}$ and $|\Lambda_i| \leq |H_i| \cdot \sqrt{n}$, this merging can be done in $O(\log n)$ time using $|\Lambda_i|$ processors in the EREW PRAM, and thus the $\sqrt{n}$ mergings, one for each h_i, can be done in $O(\log n)$ time using $O(n)$ processors in the EREW PRAM. Then for $s \in R_i$, $RTN_S(s) = t$, where $t \in \Lambda_i$ is the shortest segment taller than s, which can be easily obtained from the merged list of $R_i \cup \Lambda_i$.

Lemma 2.2 The all tallest neighbors problem can be solved in $O(\log n)$ time using $O(n)$ processors in the EREW PRAM.

2.1.2 Algorithm B: O(log n) time and O(n/log n) processors

Algorithm B here is quite similar to Algorithm A. Partition S into $n/\log n$ subsets $T_1, T_2, \cdots, T_{\frac{n}{\log n}}$, each of size $\log n$. Solve the problem for each T_i in parallel using the linear time sequential algorithm described earlier. The definitions of variables are identical as before with $n/\log n$ appropriately replacing $\sqrt{n}$. Compute $RTN_H(h)$ and $LTN_H(h)$ for each $h \in H$ by calling Algorithm A. Since we have only $n/\log n$ segments and the same number of processors, this can be performed in $O(\log n)$ time.

Computing $|H_i|$ for each i can be done by sorting. Assign $|H_i|$ processors to each pair of R_i and Λ_i and merge them by y-coordinates as follows: Make $|H_i|$ copies of R_i. To do this, let $p = |H_i|$. If $p \geq \log n$, assign $p/\log n$ processors to each entry of R_i and each entry is replicated p times in $O(\log n)$ time. If $p < \log n$, divide R_i into p blocks, each of size $\log n/p$, and each block is assigned a processor. A block can be replicated p times in $O(\log n)$ time, because a replication can be made in $O(\log n/p)$ sequential time and $p < \log n$. Assign a processor to each pair of the kth copy of R_i and $\lambda_{i,k}$ for $1 \leq k \leq |H_i|$ and merge them. This takes only $O(\log n)$ sequential time because $|R_i| \leq \log n$ and $|\lambda_{i,k}| \leq \log n$. Combining the results of these $|H_i|$ mergings will give the merged list of $R_i \cup \Lambda_i$. The remaining steps to compute $RTN_S(s)$ for each $s \in R_i$ are exactly the same as those of Algorithm A.

Theorem 2.1 If the line segments are sorted by x_i's, then the all tallest neighbors problem can be solved in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

Our solutions to the all tallest neighbors problem also solves the following problem, called the all nearest greater values problem, in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM. The problem is: Given an array $A = (a_1, a_2, ..., a_n)$ of items, to find for each i the nearest item to its right that is greater than a_i, if such an item exists. Wagener [19] and Berkman *et al.* [3] recently gave an $O(\log n)$ time, $O(n/\log n)$ processor algorithm in the CREW PRAM and the latter also gave an $O(\log\log n)$ time, $O(n/\log\log n)$ processor algorithm in the CRCW PRAM.

Theorem 2.2 The all nearest greater values problem can be solved in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

2.2 Red-Blue Matching Problem

Suppose that we have n red and blue points on the real line. We say that a red point r can be matched to a blue point b if $x(r) < x(b)$, where $x(\cdot)$ is the x-coordinate of the point. The red-blue matching problem is to find a maximum matching between the red and blue points.

Let $P = \{p_1, p_2, \cdots, p_n\}$ be the list of given points, sorted into increasing order of x-coordinates. A simple linear time algorithm works as follows: Initially, a stack $S \leftarrow \emptyset$ and $i \leftarrow 1$. If p_i is red, then ADD_S(p_i). If p_i is blue and $S \neq \emptyset$, then $p_j \leftarrow$ DELETE_S and match (p_j, p_i). Set $i \leftarrow i + 1$. Repeat until $i > n$.

To prove the correctness, let $P_i = \{p_1, p_2, \cdots, i\text{th blue point}\}$ be the prefix of P ending at the ith blue point and let M_i be the matching of P_i output by our algorithm. Then, $|M_1| = 0$ if p_1 is blue and $|M_1| = 1$ otherwise. So, M_1 is a maximum matching of P_1. Assume that M_{i-1} is a maximum matching of P_{i-1}. If P_i has more than $|M_{i-1}|$ red points, then the ith blue point is matched to the last unmatched red point in P_i by the algorithm. Thus $|M_i| = |M_{i-1}| + 1$. If P_i has exactly $|M_{i-1}|$ red points, then the algorithm clearly outputs that $|M_i| = |M_{i-1}|$. So, M_i is a maximum matching of P_i. Hence the algorithm correctly outputs a maximum matching of P. Its time complexity is clearly $O(n)$.

We shall show that if the points are initially sorted by x-coordinates, then the matching problem can be solved in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

We begin by introducing some orange and green points. If p_i is red and p_{i+1} is blue, then insert an orange point between them. If p_i is blue and p_{i+1} is red, then insert a green point between them. Let $Q = \{q_1, q_2, \cdots, q_m\}$ for $m \geq n$ be the resulting set, sorted into increasing order of x-coordinates. Assign a -1 to each of the red and orange points and a $+1$ to each of the blue and green points and apply the parallel prefix summation.

Remark: Let t_i be the prefix sum associated with q_i. Each orange (resp. green) point q_i corresponds to a local minimum (resp. maximum), i.e. $t_{i-1} > t_i < t_{i+1}$ (resp. $t_{i-1} < t_i > t_{i+1}$). For a block of red (resp. blue) points $p_i, p_{i+1}, \ldots, p_j$, the sequence $t_i, t_{i+1}, \ldots, t_j$ is decreasing (resp. increasing).

We transform our problem to an instance of the all tallest neighbors problem. Create the red vertical line segment s_i connecting two points (i, t_i) and $(i, -\infty)$ if q_i is red, and the blue segment connecting $(i, t_i + \frac{1}{2})$ and $(i, -\infty)$ if q_i is blue. No segments are created for the orange and green points. Compute $RTN(s_i)$ for each red segment s_i. Match (q_i, q_j) if $RTN(s_i) = s_j$.

Remark: If $RTN(s_i) = s_j$ for a red segment s_i, then s_j is blue. No two red segments s_i and s_j have $RTN(s_i) = RTN(s_j) = s_k$ for some blue segment s_k.

Since $|t_i|$ for a red point q_i corresponds to the height of the stack and $RTN(s_i) = s_j$ guarantees that $t_i = t_j$ and q_i is on the top of the stack when q_j is considered in our sequential algorithm, it is easy to see that the matching output by our parallel algorithm is identical to that by our sequential algorithm.

So, we have proved the following theorem.

Theorem 2.3 If the red and blue points are initially sorted by x-coordinates, then a maximum red-blue matching can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

2.3 Euler Tour Technique Revisited

The Euler tour technique of Tarjan and Vishkin [17] has many applications for computing simple tree functions such as pre-order and post-order numberings, the level or height of each vertex, and the number of descendants of each vertex. The technique computes these functions by reducing the problems to list ranking, for which $O(\log n)$ time, $O(n/\log n)$ processor EREW PRAM algorithms exist [2, 6]. We add one more to the list of applications of the technique. Suppose that we have a rooted tree T, stored in an array. Each vertex points to its parent and its first (i.e. smallest-indexed) child and all children of a vertex appear consecutively in the array. Then a vertex-to-root path in T can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

We apply the Euler tour technique to T to obtain the preorder numberings of the vertices. Since each vertex knows the positions of its children in the array, the Euler tour technique can be applied without any difficulty. Let τ be the ordering of the vertices of T in the Euler tour, starting and ending at the root, in which a vertex is represented by its preorder number. The vertex-to-root path, starting at the vertex of preorder number k, can be obtained as follows: Let τ' be the subtour of τ, starting at the last appearance of k (k may appear several times) and ending at the first appearance of 1 (the root) after the last k. Discard all integers $> k$ from τ'; let τ'' be the resulting list.

We make some observations on τ, τ' and τ''.

(i) All integers between two consecutive appearances of an integer j in τ are successors of j in T and thus $> j$.

(ii) Each integer $j < k$ in τ' is an ancestor of k.

(iii) If two integers i and j for $i < j < k$ appear in τ', then the appearances of j precede those of i in τ'.

(iv) τ'' is non-increasing and consists of the ancestors of k.

Compressing out all duplicate copies of integers from τ'' gives a decreasing sequence of integers, which is the vertex-to-root path, starting at the vertex of k. Since τ'' is non-increasing, duplicate copies can be removed from τ'' in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM by the parallel prefix computation.

Theorem 2.4 If a tree T with n vertices is stored in an array in which each vertex points to its parent and its first child and all children of each vertex appear consecutively, then the vertex-to-root path, starting at an arbitrary vertex, can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

3 Definitions

Let $A = \{a_i = \langle l_i, r_i\rangle | 1 \le i \le n\}$ be a set of intervals on the real line. The interpretation is that an interval a_i stretches from its left endpoint l_i to its right endpoint r_i. We assume that $r_i < r_{i+1}$ for $1 \le i \le n-1$. We shall use l_i and r_i to denote the left endpoint and right endpoint of a_i, resp., as well as their x-coordinates. It is often useful to transform intervals $a_i = \langle l_i, r_i\rangle$ into planar points $p(a_i) = (l_i, r_i)$ and to deal with points instead of intervals. This will be called the point representation of intervals.

We define some interval functions and in the remainder of this section we show these functions can be computed in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM if the endpoints of the intervals initially sorted.

- $NEXT(a_i)$ is the leftmost ending interval beginning after the right endpoint of a_i, i.e.

$$NEXT(a_i) = \begin{cases} a_j & \text{if } r_j = \text{MIN}\{r_k | r_i < l_k\}. \\ 0 & \text{otherwise.} \end{cases}$$

- $LMOST(a_i)$ is the leftmost ending interval containing the right endpoint of a_i, i.e.

$$LMOST(a_i) = \begin{cases} a_j & \text{if } r_j = \text{MIN}\{r_k \mid l_k < r_i < r_k\}. \\ 0 & \text{otherwise.} \end{cases}$$

- $RMOST(a_i)$ is the rightmost ending interval containing the right endpoint of a_i, i.e.

$$RMOST(a_i) = \begin{cases} a_j & \text{if } r_j = \text{MAX}\{r_k \mid l_k < r_i < r_k\}. \\ 0 & \text{otherwise.} \end{cases}$$

3.1 Computation of NEXT and RMOST

Suppose that an interval a properly contains another interval b. Then no interval c will choose $NEXT(c) = a$. So, we can discard all intervals in A that properly contain another interval in A. Note that a_i properly contains a_j if and only if $p(a_j)$ lies in the south-east quadrant of $p(a_i)$. Let P be the set of points transformed from the intervals in A. Find all points in P that have no point in its south-east quadrant. Let P' be the set of such points. Then any point $p(a_i) \in P - P'$ has a point $p(a_j) \in P'$ in its south-east quadrant, implying that a_i properly contains a_j. Thus P' corresponds to the set of intervals which are left after discarding all intervals in A that contain another interval in A. Computing P' can be done by the prefix parallel computation. We order the points in P by decreasing x-coordinates and then compute the prefix y-minimum points. The set of these prefix y-minimum points is exactly P'.

Let A' be the set of intervals corresponding to the points in P'. Let $l_{\alpha(1)}, l_{\alpha(2)}, \cdots, l_{\alpha(|A'|)}$ be the ordered list of the left endpoints of intervals in A'. Let R_i be the set of a_j's such that $l_{\alpha(i)} < r_j < l_{\alpha(i+1)}$. Then each interval $a_j \in R_i$ has $NEXT(a_j) = a_{\alpha(i+1)}$. Computing the R_i's and broadcasting $a_{\alpha(i+1)}$ to each $a_j \in R_i$ can be done by applying the parallel prefix computation.

To compute $RMOST(\cdot)$, we first make a similar observation as with $NEXT(\cdot)$. If an interval a properly contains another interval b, then no interval c will have $RMOST(c) = b$. Let A' be the subset of A after discarding all intervals from A that are contained in another interval. Order the intervals in A' by increasing left endpoints. Let $l_{\alpha(1)}, l_{\alpha(2)}, \cdots, l_{\alpha(|A'|)}$ be the sorted left endpoints of intervals in A'. If a_j is an interval with $l_{\alpha(i)} < r_j < l_{\alpha(i+1)}$, then $RMOST(a_j) = a_{\alpha(i)}$.

Remark: If all intervals in A are stored in an array of size n, ordered by increasing right endpoints, then all intervals having the same $NEXT$ appear consecutively in the array. The same is true for $RMOST$. This is a crucial property that permits some of our algorithms to work with only $O(n/\log n)$ processors.

Lemma 3.1 If the endpoints of the intervals in A are initially sorted by x-coordinates, then $NEXT(\cdot)$ and $RMOST(\cdot)$ for each interval can be computed in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

3.2 Computation of LMOST

First note that the approach used to compute $NEXT$ and $RMOST$ no longer works here. When an interval a properly contains another interval b, then we can easily find an interval c having $LMOST(c) = a$, and another interval d having $LMOST(d) = b$.

We use the point representation of intervals. It is easy to see that $LMOST(a_i) = a_j$ if $p(a_j)$ is the first point hit by the horizontal line segment connecting two points $(0, r_i)$ and (r_i, r_i) when we drag the segment vertically upwards, and $LMOST(a_i) = 0$ otherwise. To transform the computation of $LMOST(\cdot)$ to an instance of the all tallest neighbors problem, we order the points by increasing r_i's, i.e. $r_i < r_{i+1}$ and then create for each $p(a_i) = (l_i, r_i)$ two horizontal line segments, u_i, connecting (l_i, r_i) and (∞, r_i) and u'_i, connecting $(r_i, \frac{r_i + r_{i+1}}{2})$ and $(\infty, \frac{r_i + r_{i+1}}{2})$ with $r_{n+1} > r_n$. Rotating the plane 90° clockwise with respect to the point $(\infty, 0)$ gives a set $\{u_i, u'_i | 1 \leq i \leq n\}$ of $2n$ rooted vertical line segments. Then $LMOST(a_i) = a_j$ if $RTN(u'_i) = u_j$ and $LMOST(a_i) = 0$ otherwise.

Lemma 3.2 If the endpoints of the intervals in A are initially sorted by x-coordinates, then $LMOST(\cdot)$ for each interval can be computed in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

4 Algorithms

Let $A = \{a_i = \langle l_i, r_i \rangle | 1 \leq i \leq n\}$ be a set of intervals on the real line. Assume that A is connected and that $r_i < r_{i+1}$, $1 \leq i \leq n-1$. The interval a_j of A with $r_j = \text{MAX}\{r_k | 1 \leq k \leq n\}$ is called the rightmost interval of A. The interval a_j of A with $l_j = \text{MIN}\{l_k | 1 \leq k \leq n\}$ is called the leftmost interval of A.

4.1 Depth First Search

We say that a DFS tree of A is R-restricted (resp. L-restricted) if its starting interval is the rightmost (resp. leftmost) interval of A. Let $T = (V, E')$ where $V = A$ and $E' = \{(a_i, LMOST(a_i)) | 1 \leq i \leq n-1\}$.

Lemma 4.1 T is an R-restricted DFS tree of A.

Proof. See Moitra and Johnson [15]. □

Corollary 4.1 If the endpoints of the intervals in A are initially sorted by x-coordinates, then an R- or L-restricted DFS tree can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

Proof. For L-restricted DFS trees, we use $LMOST'(\cdot)$, instead of $LMOST(\cdot)$, which is defined as follows:

$$LMOST'(a_i) = \begin{cases} a_j & \text{if } l_j = \text{MAX}\{l_k | l_k < l_i < r_k\}. \\ 0 & \text{otherwise.} \end{cases}$$

□

We shall show how we make use of the restricted DFS tree algorithm to develop an algorithm for computing a DFS tree with an arbitrary starting interval a_s.

Choose an arbitrary point p on a_s and let B be the set of intervals containing p. Let $C_1, C_2, \cdots, C_m$, $m > 0$, be the connected components of $A - B$ in left-to-right order. We first construct a path $P = (a_{b_1}, a_{b_2}, \cdots, a_{b_k})$, $b_1 = s$, which consists of the intervals in B (in an arbitrary order). We say that a_{b_i}, $1 \leq i \leq k$, and C_j, $1 \leq j \leq m$, intersect if a_{b_i} intersects any interval in C_j. For each C_j, find the largest index $con(j)$ such that $a_{b_{con(j)}}$ and C_j intersect. If C_j is on the left of p, then construct an R-restricted DFS tree of C_j and link it as a subtree of $a_{b_{con(j)}}$. Otherwise, construct an L-restricted DFS tree and link it. Let T be the resulting tree.

Lemma 4.2 T is a DFS tree with starting interval a_s.

Proof. Omitted. □

Theorem 4.1 If the endpoints of the intervals in A are initially sorted by x-coordinates, then a DFS tree with an arbitrary starting interval can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

Proof. Omitted. □

4.2 Breadth First Search

Let $T = (V, E')$ where $V = A$ and $E' = \{(a_i, RMOST(a_i)) | 1 \le i \le n-1\}$.

Lemma 4.3 T is a BFS tree of A with starting interval being the rightmost interval of A.

Proof. Omitted. □

Corollary 4.2 Let T_{a_i} be the subtree of T rooted at a_i. Then T_{a_i} is a BFS tree, with starting interval a_i, of the intervals which are the descendants of a_i

We shall develop an algorithm for computing a BFS tree with an arbitrary starting interval a_s. Let B be the set of intervals intersecting a_s. Define

$$Q(a_i) = \begin{cases} 0 & \text{if } a_i = a_s, \\ a_s & \text{if } a_i \in B, \\ RMOST(a_i) & \text{if } a_i \text{ lies entirely to the left of } a_s, \\ RMOST'(a_i) & \text{otherwise,} \end{cases}$$

where

$$RMOST'(a_i) = \begin{cases} a_j & \text{if } l_j = \text{MIN}\{l_k | l_k < l_i < r_k\}. \\ 0 & \text{otherwise.} \end{cases}$$

Let T be the tree defined by $Q(\cdot)$. The root of T is a_s, the intervals in B are the children of a_s and the intervals in $A - B - \{a_s\}$ are of depth ≥ 2. The depth of the root is 0.

Lemma 4.4 T is a BFS tree with starting interval a_s.

Proof. Omitted. □

Theorem 4.2 If the endpoints of the intervals in A are initially sorted by x-coordinates, then a BFS tree with an arbitrary starting interval can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

4.3 Maximum Independent Set

A subset I of A is an independent set if no two intervals in A intersect. A maximum independent set is an independent set with maximum cardinality.

The graph defined by $NEXT(\cdot)$ is a forest of trees. So, we introduce an interval a_{n+1} with $l_{n+1} = \infty$. Then the graph becomes a tree rooted at a_{n+1}. It is easy to see that the subset I defined inductively: $a_1 \in I$; if $a_i \in I$, then $NEXT(a_i) \in I$, is a maximum independent set of $A \cup \{a_{n+1}\}$. Compute a maximum independent set of $A \cup \{a_{n+1}\}$ by finding the vertex-to-root path, starting at the vertex corresponding to a_1 by the Euler tour technique. Then $I - \{a_{n+1}\}$ is a maximum independent set of A.

Theorem 4.3 If the endpoints of the intervals of A are initially sorted by x-coordinates, then a maximum independent set of A can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

4.4 Minimum Clique Cover

A partition $C_1, C_2, ..., C_m$, $m > 0$, of A is a clique cover if each C_i is a clique, i.e. every two intervals in C_i intersect. A minimum clique cover is a clique cover with minimum m.

Let $I = \{a_{\beta(i)} | 1 \leq i \leq |I|\}$ be a maximum independent set of A output by our maximum independent set algorithm in the last section. Let $r_{\beta(0)} = -\infty$. It is easy to see that a clique cover $C_1, C_2, \cdots, C_{|I|}$ is minimum, where $C_i = \{a_j | r_{\beta(i-1)} < l_j \leq r_{\beta(i)}\}$. Each C_i is a clique. Each a_j belongs to exactly one C_i. We need at least $|I|$ cliques to cover A. Computing the C_i is straightforward.

Theorem 4.4 If the endpoints of the intervals of A are initially sorted by x-coordinates, then a minimum clique cover of A can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

4.5 Minimum Dominating Set

A subset D of A is a dominating set if each interval in $A - D$ intersects an interval in D. A minimum dominating set is a dominating set with minimum cardinality.

Assume that all intervals in A are stored in an array, ordered by their right endpoints and that each interval has its $NEXT$ and $RMOST$. Let $RTNXT(\cdot) = RMOST(NEXT(\cdot))$. We can easily verify that all intervals having the same $RTNXT$ appear consecutively in the array. Since the graph defined by $RTNXT(\cdot)$ may consist of several trees, we also introduce a_{n+1} as in the maximum independent set problem. It is easy to see that the subset D of A defined inductively: $RMOST(a_1) \in D$; if $a_i \in D$ then $RTNXT(a_i) \in D$, is a minimum dominating set of $A \cup \{a_{n+1}\}$. We use the Euler tour technique to compute a minimum dominating set D of $A \cup \{a_{n+1}\}$. Then $D - \{a_{n+1}\}$ is a minimum dominating set of A.

Theorem 4.5 If the endpoints of the intervals of A are initially sorted by x-coordinates, then a minimum dominating set of A can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

4.6 Minimum Coloring

A coloring of A is to assign a color to each interval in A so that no two intersecting intervals have the same color. A minimum coloring is a coloring with a minimum number of colors.

Let P be the set of endpoints of the intervals in A. Assume that the right (resp. left) endpoints are red (resp. blue). Compute a maximum red-blue matching M of P. Link the interval with right endpoint r to the interval with left endpoint b for each matched pair $(r, b) \in M$. Let $L \subseteq A$ be the set of intervals whose left endpoints are unmatched in M. Then A is partitioned into $|L|$ lists, each of which is an independent set, headed by an interval in L. So, A can be colored with $|L|$ colors. A coloring of A with less than $|L|$ colors immediately implies a red-blue matching of size $> |M|$. Thus $|L|$ colors are necessary and sufficient to color A.

We have partitioned A into $|L|$ lists. To color each interval in A, assign color i to the ith interval of L for $1 \leq i \leq |L|$ and apply the extended list ranking of Cole and Vishkin [7]. In the extended list ranking problem, there are more than one list and the problem is to compute the rank of each entry in its own list. At the completion of applying the extended list ranking, each interval in the ith list is assigned color i.

Theorem 4.6 If the endpoints of the intervals of A are initially sorted by x-coordinates, then a minimum coloring of A can be found in $O(\log n)$ time using $O(n/\log n)$ processors in the EREW PRAM.

Acknowledgement. The author wishes to thank Professor W.L. Ruzzo for his helpful discussions and his careful reading of an earlier draft of this paper.

References

[1] M.J. Atallah and M.T. Goodrich, Efficient parallel solutions to some geometric problems, *Journal of Parallel and Dist. Comput.*, **3** (1986), 492–507.

[2] R.J. Anderson and G.L. Miller, Deterministic parallel list ranking, *Proc. 3rd Aegean Workshop on Computing*, 1988, pp. 81–90.

[3] O. Berkman, D. Breslauer, Z. Galil, B. Schieber and U. Vishkin, Highly parallelizable problems, *Proc. ACM Symp. on Theory of Computing*, 309–319 (1989).

[4] A.A. Bertossi, Finding Hamiltonian circuits in proper interval graphs, *Information Processing Letters*, **17** (1983), 97–101.

[5] A.A. Bertossi and M.A. Bonuccelli, Some parallel algorithms on interval graphs, *Discrete Applied Mathematics*, **16** (1987), 101–111.

[6] R. Cole and U. Vishkin, Deterministic coin tossing with applications to optimal parallel list ranking, *Information and Control*, **70** (1986), 32–53.

[7] R. Cole and U. Vishkin, The accelerated centroid decomposition technique for optimal parallel tree evaluation in logarithmic time, *Algorithmica*, **3** (1988), 329–346.

[8] M.C. Golumbic, Algorithmic graph theory and perfect graphs, Academic Press, New York, 1980.

[9] U.I. Gupta, D.T. Lee and J.Y.T. Leung, An optimal solution for the channel assignment problem, *IEEE Trans. Computers*, **C-28** (1979), 807–810.

[10] U.I. Gupta, D.T. Lee and J.Y.T. Leung, Efficient algorithms for interval graphs and circular-arc graphs, *Networks*, **12** (1982), 459–467.

[11] M.J. Keil, Finding Hamiltonian circuits in interval graphs, *Information Processing Letters*, **20** (1985), 201–206.

[12] C.P. Kruskal, L. Rudloph and M. Snir, The power of parallel prefix, *IEEE Trans. Computers*, **C-34** (1985), 965–968.

[13] R. Ladner and M. Fischer, Parallel prefix computation, *Journal of ACM*, **27** (1980), 831–838.

[14] G.K. Manacher and T.A. Mankus, An optimum $\theta(n \log n)$ algorithm for finding a canonical Hamiltonian path and a canonical Hamiltonian circuit in a set of intervals, *Proc. Annual Allerton Conf. Communication, Control and Computing*, 1987, pp. 24–33.

[15] A. Moitra and R. Johnson, Parallel algorithms for maximum matching and other problems on interval graphs, Tech Report 88-927, Dept of Computer Science, Cornell University, 1988.

[16] J.E. Savage and M.G. Wloka, A parallel algorithm for channel routing, Tech Report CS-88-11, Computer Science Dept., Brown University, 1988, *Revised in Jan. 1989*.

[17] R.E. Tarjan and U. Vishkin, An efficient parallel connectivity algorithm, *SIAM J. Comput.*, **14** (1985), 862–874.

[18] L.G. Valiant, Parallelism in comparison problems, *SIAM J. Comput.*, **4** (1975), 348–355.

[19] H. Wagener, Triangulating a monotone polygon in parallel, *Proc. Int'l Workshop on Computational Geom.* (Lecture Notes in Computer Science. Vol. 333) 136–147 (1988).

AN NC ALGORITHM FOR FINDING A MAXIMUM WEIGHT CLIQUE ON A CIRCLE GRAPH

C. RHEE, S. K. DHALL and S. LAKSHMIVARAHAN
Parallel Processing Institute
School of Electrical Engineering and Computer Science
University of Oklahoma, Norman, Oklahoma

ABSTRACT

In this paper, we show that the maximum weight clique problem for a circle graph can be solved in $O(\log n)$ time using polynomially many processors in the CRCW parallel machine model.

1. INTRODUCTION

Let S be a set and $C = \{S_1, S_2, ..., S_m\}$ be a non-empty collection of distinct non-empty subsets of S whose union is S. A graph $G = (V, E)$ is called an *intersection graph* for C if there is one-to-one correspondence between C and V such that $S_i \cap S_j \neq \varnothing$, $i \neq j$, iff $v_i v_j \in E$, where v_i and v_j are vertices in V corresponding to S_i and S_j, respectively. When C is a collection of chords on a circle, G is called a *circle graph* for S. Let $< n > = \{1, 2, ..., n\}$ and $\Pi = (\Pi(1)\, \Pi(2) \cdots \Pi(n))$ be a permutation on $< n >$. Construct an undirected graph $G(\Pi) = (< n >, E)$, where $(i, j) \in E$ if and only if $(i - j)(\Pi^{-1}(i) - \Pi^{-1}(j)) < 0$. In other words, there is an edge connecting nodes i and j if and only if $(\Pi(i), \Pi(j))$ is an inversion in Π. Any undirected graph G is called a *permutation* graph if there exists a Π such that G is isomorphic to $G(\Pi)$. The permutation graph $G = (V, E)$ will be denoted as $G = (\Pi(1)\Pi(2) \cdots \Pi(n))$ in this paper, where Π is the underlying permutation for G.

Let $G = (V, E)$ be an unweighted graph and V' be a subset of V. Then the subgraph induced by V' is *clique* if it is complete. A clique Q is called a *maximum cardinality clique* of G if the vertex set of Q has the largest cardinality among all cliques. A graph G is called a *weighted* graph if there exists a weight function $w : V \rightarrow R$ (the set of reals) where $w(u)$ is called the weight of node u. Let V' be a subset of V and $S(V') = \sum_{u \in V'} w(u)$. Then a clique Q^* of G such that $S(V(Q^*)) = \max_{Q}\{S(V(Q))\}$, where Q is a clique and $V(Q)$ is the vertex set of Q, is called a maximum weight cilque for G.

Though the problem of finding a maximum clique for an arbitrary graph is known to be *NP*-complete [1], this problem can be solved in $O(n^3)$ time for the case of weighted circle graphs [2]. Recently, Hsu [3] reduced the time complexity of the problem to $O(n^2 + m \log\log n)$. For the maximum cardinality clique problem, Buckingham [4], Rotem and Urrutia [5] have given $O(n^2)$ algorithms. This paper shows that the maximum weight clique problem for the circle graph is in *NC* class by presenting an $O(\log n)$ time parallel algorithm using polynomially many processors.

2. THE ALGORITHM

Consider a set S of n chords of a circle C, where each chord is denoted by two endpoints ab (or equivalently (a, b)), $a < b$, and the endpoints of the chords are numbered from 1 to $2n$, clockwise around the circle. (See Fig-1.) We assume without loss of generality that no two chords of S share an endpoint, since we can replace any common endpoint by more than one consecutive points on C without changing the circle graph [2]. For the given chord set S, our algorithm reads S as its input and outputs a maximum weight clique for S.

Let $S_i = \{ ab \in S \mid a \leq i < b \}$, for each $i = 1, 2, ..., 2n$. From Fig-1, we have $S_1 = \{ (1, 13) \}$, $S_2 = \{ (1, 13), (2, 9) \}$, $S_6 = \{ (1, 13), (2, 9), (3, 16), (4, 7), (5, 8), (6, 12) \}$, $S_{15} = \{ (3, 16) \}$, $S16 = \varnothing$, etc.. Our algorithm is based on the following lemmas.

LEMMA 1 [6]: Each S_i realizes a permutation graph.

LEMMA 2 [6]: Let Q be a subset of S. Then Q is a clique for S iff there is S_i for which Q is also a clique.

LEMMA 3: The maximum weight clique problem for the permutation graph can be solved in $O(\log n)$ time using $O(n^{3+\epsilon})$ processors in the CRCW parallel model.

Proof: Let $G = (V, E)$ be a weighted permutation graph, where nonnegative weight is assigned to each node in V. We construct a weighted digraph $H = (V^H, E^H)$ as follows: Let $V^H = V$. For each $i, j \in V$ $(i, j) \in E^H$ with

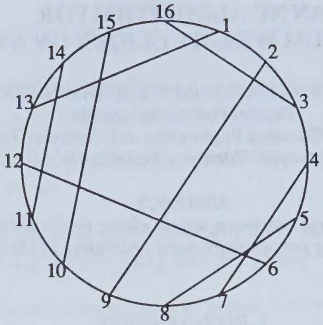


Fig-1. A Circle Graph.

weight $-w(i)$ iff $i < j$ and $\Pi^{-1}_i > \Pi^{-1}_j$. Clearly, the digraph H is acyclic. It is not difficult to show that any maximam clique for G corresponds to a shortest directed path in H. The construction of H can be done in a constant time using $O(n^2)$ processors. Finally, applying a known *NC* algorithm for finding a shortest path in a graph with no negative cycle [7], we can derive an $O(log\ n)$ time algorithm with $O(n^{3+\epsilon})$ processors for finding a maximum weight clique for G.

THEOREM 1 : The maximum weight clique problem for the circle graph can be solved in $O(log\ n)$ time using polynomially many processors in the CRCW model.

Proof: Firstly, each S_i can be constructed in a unit of time. Then, after computing a maximum weight clique for each S_i from Lemma 3, we can simply pick up a maximum weight clique for S out of $2n$ such cliques.

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and intractability*, W. H. Freeman and Company, New York, 1979.

[2] F. Gavril, "Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle Graph", Networks, Vol 3, 1973, pp. 261 - 273.

[3] W. L. Hsu, "Maximum Weight Clique Algorithms for Circular-arc Graphs and Circle Graphs", SIAM J. Comput., Vol 14 - 1, 1985, pp. 224 - 231.

[4] Buckingham, "Efficient Stable Set and Clique Finding Algorithms for Overlap Graph", Dept. Computer Science, New York Univ., New York, 1981.

[5] D. Rotem and J. Urrutia, "Finding Maximum Cliques in Circle Graphs", Networks, Vol 11, 1981, pp. 269 - 278.

[6] M. C. Golumbic, *Algorithmic Graph Theory* and *Perfect Graphs*, Academic Press, New York, 1980.

[7] R. C. Paige and C. P. Kruskal, "Parallel Algorithms for Shortest Path Problems", 26th IEEE FOCS, 1985, pp 14 - 20.

PARALLEL GARBAGE COLLECTORS FOR THE RP3*

PETER REVESZ

Department of Computer Science
Brown University
Providence, Rhode Island 02912

Abstract: This paper presents several parallel garbage collectors that are suitable for execution on shared memory multiprocessors like the RP3 that have efficient implementation of the Fetch-and-Add operation. We analyze the algorithms first theoretically then give estimated speed-ups for cases when the number of nodes in main memory is much greater than the number of processors. We also describe a new parallel transitive closure algorithm for directed graphs that runs in $O(nlogn)$ time with $O(max(n,e))$ processors and a single source transitive closure algorithm that runs in $O(\sqrt{n}logn)$ time with $O(\sqrt{n}max(n,e))$ processors, where n is the number of vertices and e is the number of edges.

1 INTRODUCTION

With the advance of larger and increasingly parallel list processing systems, there is a growing need for fast automatic storage reclamation or garbage collection. For example, future list processing systems such as graph-reducers (ex. [14],[20]) could harness in principle the implicit parallelism among reductions that emerges naturally from their Church-Rosser property. Parallelizing the garbage collector seems an attractive and common answer, which in our case the development of the RP3 (for Research in Parallel Processing Prototype) computer greatly encouraged.

Garbage collection is a memory management technique for list processing systems that use linked data structures with single nodes as building blocks, assume one special node called the root, and perform operations only on the descendants of the root. The list processor, while performing its operations, may add or delete links and make nodes non-descendants of the root. Since no more operations can be done on these non-descendants of the root, they are called garbage. The purpose of garbage collection is to recycle garbage nodes by finding them and linking them in a free list while allowing the list processing system to retrieve nodes form the free list. Following the terminology of [9] the list processing system without the garbage collector is called the *mutator*.

Garbage collectors can be either ***stop-and-collect***, i.e. require the mutator to wait for the garbage collector, or ***on-the-fly***, i.e. allow concurrent execution of the two. The first on-the-fly garbage collector using semaphores was written by Steele [25], while the first on-the-fly garbage collector without semaphores is due to Dijkstra et al. [9]. Kung and Song [15] gave a more efficient garbage collector with one dedicated processor. Wadler [29] considered garbage collectors with several dedicated processors. Halstead also gave a parallel stop-and-collect garbage collector based on the incremental garbage collector of Baker. Hickey and Cohen [13] analyzed on-the-fly garbage collectors and gave conditions for equilibrium or stability. Ben-Ari [4] described several new parallel garbage collectors including one incremental. For more information see Cohen's excellent survey [7] on garbage collection.

The parallel garbage collectors presented in this paper have the following distinguishing features.

1. They execute stop-and-collect.
2. They use a large number of processors with almost linear speed-ups.
3. They do not collect all garbage after a fixed number of cycles.

We chose (1) to require the minimum of adaptation and cooperation from the mutator, as in on-the-fly garbage collection the mutator also has to mark nodes. We chose (2) to allow a high degree of parallelism to be limited by chiefly only the available hardware. In fact, we felt so strongly about (2) that we loosened the usual requirement of the negation of (3), although we did try to satisfy that also and during our trial we developed two new (single source) transitive closure algorithms.

---

*Most of the research presented here was done during a summer visit at the IBM T.J. Watson Research Center.

2 THE RESEARCH PARALLEL PROCESSING PROTOTYPE

The Research Parallel Processing Prototype (RP3) [18] under development at IBM in cooperation with the NYU Ultracomputer project is a highly parallel computer consisting of 512 Processor-Memory Elements (PMEs). Each PME contains an IBM 801-like microprocessor, memory-mapping unit, 32K-byte cache, 4 Mbytes of memory, an I/O interface, and an interface to a high-performance interconnection network. The RP3 supports shared memory for the PMEs.

The RP3's MIMD feature makes it a natural choice for highly parallel list processing, although our algorithms can be executed SIMD style.

The main synchronization operation provided in the RP3 is the fetch-and-add operation, a nonblocking primitive that can be performed in one memory cycle. Syntactically an (integer) fetch-and-add operation is $ifna(\&shloc, locval)$, which results in adding the value of a local variable *locval* to the shared location *&shloc* and incrementing by one the value of the shared variable shloc. The fetch-and-add operation is frequently used for insertion and deletion from a shared queue, when it is implemented as a set of consecutive shared memory locations. We exploit this feature, although our algorithms are implementable on other machines.

The RP3 interconnection network, also called Ω-network, supports message combining. In particular, the cost of simultaneous fetch-and-add operations to the same location is logarithmic in the number of processors [11].

Our algorithms were implemented in the IBM EPEX/C language, an extension of C suitable for expressing parallelism in a shared memory environment.

3 APPROXIMATE ALGORITHMS USING CONNECTIVITY

Our strategy for garbage collection can be best described as greedy. We collect in each phase all the garbage that we can collect fast easily, leaving hard-to-collect garbage for later phases. In particular, our algorithms are based on the highly parallel recognizability of two types of garbage: *orphan* nodes, which have no parents, and *disjoint* nodes, which have no path to them from the root even in the undirected version of the graph. Since some garbage can be neither orphan nor disjoint, the garbage collector that we describe in this section is only approximate for each run. However, a garbage node which is not collected in one phase may be collected in a later phase because *its classification can change to orphan or to disjoint* as a result of either mutator or garbage collector activity.[1]

For example, in Figure 1 assuming that node 1 is the root the node 8 is an orphan and node 9 is a disjoint garbage. Notice that the garbage structure with nodes 2, 3, and 4 is not easily collectible, because it has no orphans and it is joint to the main component. However, it may become disjoint if the data structure changes such that the edge from 1 to 5 is deleted.

By highly parallel recognizability we mean algorithms that run in polylogarithmic time with a linear number of processors in the number of vertices and edges. The class of problems solvable by such algorithms is called *resource-effective*, which is of course a restriction of the class NC. Fortunately, a large set of graph problems ranging from finding minimum spanning trees [3], biconnected components [27], triconnected components [17], euler tours [2], and embeddings of planar graphs [16] to finding connected components have this property. While the problem of finding all garbage nodes is not known to be resource-effective, a good approximate algorithm can be based in part on solutions to the above problems.

Our parallel garbage collector has the following outline. First, it identifies orphans and deletes their pointers to children. Second, it builds the undirected version of the graph by creating a list of undirected edges. Third, it finds the connected components of the undirected graph. Fourth, it marks the non-disjoint nodes. Finally, it collects all unmarked nodes.

The effect of the first step is to make all orphans disjoint. The simple method is to use one extra bit field, or tag, per node and mark the root and the children of each node. (We mark the root just in case it is an orphan because the root we do not want to collect.) Then we delete the children of unmarked nodes and reset the tags in one scan of memory. With a linear number of processors the time is $O(logn)$ as n concurrent write operations take that long in the RP3.

The second step is done using the special fetch-and-add RP3 facility, and an edge queue as mentioned above. For each directed edge a request for adding to the queue is performed using fetch-and-add. This list of edges is a temporary scratch-pad for the next step, which freely deletes from it. This step is also $O(logn)$ in the RP3.

---

[1] This is better than reference counting because reference counters can never hope to collect cyclic garbage, which can never change to acyclic.

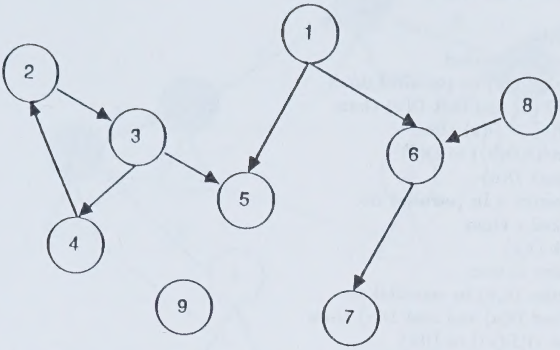


Figure 1: Example data structure

To find disjoint nodes several parallel connectivity algorithms can be used, which include the algorithms of Hirschberg, Chandra, and Sarwate [12], Chin, Lam and Chen [6], Shiloach and Vishkin [24], Awerbuch and Shiloach [3], Vishkin [28], Reif [19], Cole and Vishkin [8], and Gazit [10]. We implemented two of the connectivity algorithms, namely Shiloach and Vishkin's and Reif's (for details see [23]), and we also wrote our own which we describe in the next section.

The fourth and the fifth steps can be done together as each node can check whether it is in the same connected component as the root is. If not, then its processor by a fetch-and-add request adds it to the free list queue. (The free list queue and its head and tail pointers are shared variables.)

There are three modifications to the above outlined method that we mention:

1. The space efficiency of the parallel garbage collector is increased by the following modification. Let the processors find the undirected edges during the algorithm by repeatedly scanning the list of memory nodes during each marking phase. For example, when each node has two children pointers the vertex processors can also act as edge processors by checking whether the pointers are real.

2. During a long computation the parallel garbage collector can be occasionally substituted by a slower method like [9] to ensure collection of all garbage. This was suggested in [21] in connection with a garbage collector for orphan nodes, and later implementation by Janice Stone in [26] shows it to be a practical idea.

3. Our parallel garbage collector can run on-the-fly by requiring the mutator, as other on-the-fly garbage collectors do, to mark a node when a pointer to it is deleted, an by collecting only those connected components which have neither the root nor any marked node in them.

4 A PARALLEL CONNECTIVITY ALGORITHM

Our parallel connected components algorithm was designed to reduce the space requirement found in other algorithms. Our algorithm has an $O(log^2 n)$ worst-case running time, only one tag and one word space requirement per node, and a relatively simple proof. Hence it compares favorably with Reif's algorithm, which runs only in expected $O(log^2 n)$ time and has an implicit space requirement that each processor has access to a random number generator. It also compares favorably with Shiloach and Vishkin's algorithm, which needs three extra words per node and twelve lemmas to prove correctness.

The main data structure of the three algorithms is a reverse-rooted forest of nodes, which is stored using for each node v a pointer field $D(v)$ that points to its parent or to itself. Each reverse-rooted tree is commonly called a *supervertex*, and a depth-one supervertex is also called a *star*. The depth of a supervertex is halved by the shortcut operation which substitutes grandparents for parents, i.e. sets $D(v)$ to $D(D(v))$ for each node v. Intuitively, supervertices represent different connected components, and upon finding an edge connecting nodes in two distinct supervertices the latter are united.

Technically, the union operation in our algorithm (see Figure 2) is performed for each edge (i, j) when $D(i)$ and $D(j)$ are distinct roots (of supervertices) by a *hooking* operation which makes one of the two

```
Set parent D(v) to v for each vertex v.
Repeat
        Set live to false.
        Set vertices to unmarked.
        For every edge(u,v) in parallel do
            If root D(u) and root D(v) then
              If $D(u) > D(v)$ then
                set D(D(u)) to D(v)
                mark D(u)
        For every vertex v in parallel do
            If marked v then
              mark D(v)
              set live to true
        For every edge (u,v) in parallel do
            If marked D(u) and root D(v) then
                set D(D(v)) to D(u)
        For every vertex v in parallel do
            set D(v) to D(D(v))
Until live.
```

Figure 2: Connectivity algorithm

roots a parent of the other. Hence our algorithm is different from Shiloach and Vishkin's algorithm, which allows hooking roots to nonroots and requires a complicated checking for supervertex stagnancy. The union operations must be done carefully to avoid creating cycles. Therefore we break the union operations into two sets that are performed in sequence. In the first set each root becomes a parent of some other root with a smaller ID number. (When there are several possible roots to hook to an arbitrary one succeeds.) Then we mark all unioned nodes. In the second set each unmarked root becomes a parent of some other marked root. Then we do a shortcut operation and reset the marker of each node.

For example, in Figure 3 the supervertices with roots 2, 5, 7, and 9 can form a tree in step 1, and the supervertices with roots 3 and 4 can join it as leaves, assuming that the dashed edges are in the graph with both endpoints near the supervertex roots.

We can prove the following:

Theorem 4.1. *The above algorithm computes the connected components of an undirected graph in $O(logn)$ iterations with $O(max(n, e))$ processors.*

Proof: To show correctness, note that in each iteration the union operation is performed only between nodes that were roots at the beginning of that iteration. The first set of unions avoids cycles by the monotonicity and it creates a forest of supervertices. The second set of unions only adds leaves to the freshly created trees and therefore keeps their tree structures.

To show running time, consider the nodes in any connected component. Suppose that there are n nodes. Then we would like to perform $n-1$ union operations in $O(logn)$ iterations. Before each iteration divide the supervertices into three groups: (1) true supervertices, (2) sleeping stars, which are not unioned, and (3) live stars, which are unioned during the iteration.

A supervertex created (augmented) by k unions during one iteration will become a star within $O(logk)$ (extra) iterations. Therefore it needs not be unioned until it becomes a star. A sleeping star is waiting for a supervertex to become a star. When the supervertex becomes a star, then all sleeping stars can be added to it in one iteration. Finally, the number of live stars will be at least halved, as each is unioned with another live live star or a true supervertex.

5 ANALYSIS

Similarly to Hickey and Cohen [13] in our theoretical analysis we make a few assumptions about the list processing system, namely:

1. During the list processing the number of descendants of the root is a constant L.

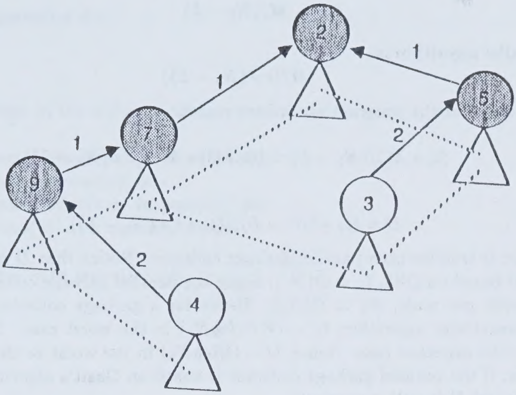


Figure 3: Example for connectivity

2. The percentage of garbage nodes collected by the parallel algorithm is some constant c after any run of the garbage collector.[2]
3. Using several processors in the parallel garbage collector yields an optimal speed-up.

Hickey and Cohen make similar assumptions to (1) and (2), while we introduce (3) to consider, unlike they, several processors. In the following analysis we consider a practical system in which there are at most two children per node and a bounded number of parents. In our analysis we highlight a concept called *dead factor*. The dead factor of a parallel garbage collector is the theoretical number of processors needed with the above assumptions to match the speed of the fastest sequential garbage collector. Intuitively, when the dead factor is D, and we want an x times faster garbage collector, we need xD processors. Therefore it is justified to call it a factor, and it is important to minimize it. For the "fastest sequential garbage collector" we chose a common stop-and-collect garbage collector which uses an iterative version of the depth-first-search algorithm in [1] to mark the descendants of a given root. Henceforth we will refer to this algorithm as DFS.

Since the parallel algorithm has different space requirements per node, the available main memory will be divided into different size nodes. (DFS needs one boolean tag and upto one extra word of stack space per node. Our most space efficient parallel algorithm needs one boolean tag and one extra pointer per node, and it needs space for a free list queue.) Hence let the number of nodes in main memory be N_1 for DFS and N_2 for the parallel algorithm. Furthermore, let the (sequential) running time for each run be t_1 for DFS and t_2 for the parallel algorithm. Finally, let M be the number of nodes needed by the list processing system during the execution of a program.

Now we can calculate the dead factor as follows. The number of necessary runs for the garbage collector based on depth-first-search is:

---

[2] This assumption in turn rests on the following. There are three types of nodes before any run of the garbage collector: L – descendants of the root, F – collectible garbage and G – uncollectible garbage. In addition, by our previous assumption that L is constant, F + G is constant, and the number of F is also the number of free list nodes before and the number of new garbage nodes after the next run of the list processing system. Then assuming that a fixed percent p of the new garbage nodes becomes uncollectible, and also a fixed percent q of G becomes collectible by the next cycle, we have

$$\begin{bmatrix} F_{k+1} \\ G_{k+1} \end{bmatrix} = \begin{bmatrix} 1-p & q \\ p & 1-q \end{bmatrix} \begin{bmatrix} F_k \\ G_k \end{bmatrix}$$

When this system reaches equilibrium, c will be:

$$c = F/(G+F) = q/(p+q)$$

$$M/(N_1 - L)$$

and for the parallel algorithm is:

$$M/(c * (N_2 - L))$$

Then when the speed of the program executions match:

$$(t_1 * M)/(N_1 - L) = ((t_2/D) * M)/(c * (N_2 - L))$$

Hence D is

$$D = (t_2 * (N_1 - L))/(t_1 * c * (N_2 - L))$$

The last equation is true for each parallel garbage collector. Notice that D need not be constant. For the garbage collector based on DFS, $t_1 = O(N_1)$. Since the parallel garbage collectors use only a constant number of extra words per node, $N_2 = O(N_1)$. Hence for a garbage collector based on Shiloach and Vishkin's or our connectivity algorithm, $t_2 = O(N_1 log N_1)$ in the worst case. For Reif's algorithm also $t_2 = O(N_1 log N_1)$ in the expected case. Hence $D = O(log N_1)$ in the worst or the expected case for these algorithms. However, if the parallel garbage collector is based on Gazit's algorithm [10], then $t_2 = O(t_1)$ in the expected case and D is only a constant.

6 TESTING RESULTS

We tested the garbage collectors at first in parallel using simulations of EPEX/C as the RP3 was still under development. Then we tried them with one processor and disregarding processor scheduling. Before this we made the algorithms sequential by simply changing the parfor control statements to for statements, and deleting the extra keywords shared and private from the declaration of variables. No other changes were made to the algorithms. Then we tested each algorithm on random data structures of increasing sizes from 100 nodes to 100000 nodes, and measured their CPU running times in microseconds on an IBM 3081 mainframe (see table).

For 100 nodes				
algorithm	marking	collecting	garbage	cycle
dfs	394	94	7	1
new	2219	216	7	3
sv	4774	216	7	5
reif	11946	216	7	17

For 1000 nodes				
algorithm	marking	collecting	garbage	cycle
dfs	2115	1199	333	1
new	34901	1891	333	4
sv	57296	1891	333	7
reif	397530	1891	333	21

For 10000 nodes				
algorithm	marking	collecting	garbage	cycle
dfg	5063	19086	8445	1
new	391680	11381	8445	5
sv	846814	11381	8445	10
reif	1893782	11381	8445	21

For 100000 nodes				
algorithm	marking	collecting	garbage	cycle
dfs	157192	144576	52446	1
new	5356123	165037	52446	6
sv	11512499	165037	52446	12
reif	25278184	165037	52446	36

Each of the data tested was generated randomly, by using a random permutation of the numbers form 1 to K, where K was the size of the memory, and then assigning the ith node the ith number in the

```
For every vertex v in parallel do
    Set v[] to false.
    Set v[v] to true.
    Set vaux[] to zero.
    Make v the only item in the queue of v.
Repeat $2n$ times
        For every directed edge(v,u) in parallel do
            write u to vaux[head(u)] of v
        For every directed edge(v,u) in parallel do
            If vaux[head(u)] of v equals u then
                pass(v,u)
        For every vertex v in parallel do
            pop(v)
Until
```

Figure 4: Transitive closure algorithm

permutation as the left child. The data structures so generated were a collection of cycles, out of which all but one was garbage. There were no orphan nodes in these data structures, so we knew that each garbage collector must collect the same amount of garbage. We took such a simple data structure, because it was easy to extend and gives meaningful comparison between the running times for different sizes.

For each of the garbage collectors we measured the marking time and the collecting times separately. For each size of memory we found that the garbage collector that is based on depth-first-search was the fastest, the new algorithm was second, sv was third, and reif was the slowest in our implementations. The last three of these algorithms used the same collector. We counted the edgelist creation and collecting times separately. The table indicates that in the RP3, if optimal speed-ups are attainable, then garbage collection using the new algorithm and 512 processors can be about $(157192 + 144576)/((5356123 + 165037)/512) = 28$ times faster than with an algorithm using depth-first-search. The actual speed-up depends largely on the processor scheduling, which we discuss next.

7 PROCESSOR SCHEDULING

One approach to processor scheduling when both the number of memory nodes and the number of processors is known in advance is to divide the number of memory nodes into the number of processors and assign the same number of nodes to each processor during the the parallel for loops. The advantage of this approach is that it is simple to implement and requires only a small overhead. The disadvantage is that it does not take care of load balancing, i.e. it could make some processors wait idle while others are still working. There is reason to believe, however, that the algorithms described will not give an unbalanced situation easily. This is because each step of the for loops consists of a few instructions only, and the processors all have to join before going to the next for loop.

8 A MORE GENERAL ALGORITHM

So far we considered only the problem of finding the descendants of a single root node. Since many list processing systems assume several roots, in this section we describe a more general algorithm that finds the descendants of each node. This is known as the transitive closure problem and is closely related to matrix multiplication, whose complexity is unresolved [1]. Here (as in [23]) we exploit the fact that often the number of edges is small, and for sparse graphs our algorithms seem better theoretically than the known general methods.

Our algorithm (see Figure 4) assumes the following data structures per node:

1. A boolean vector of length n. We denote the vector of node k by $k[]$. Initially only the entry $k[k]$ is true. When the algorithm terminates, the entry $k[i]$ is true if and only if node i is a descendant of node k. We assume that every node is a descendant of itself.
2. A queue. During the algorithm the integer i is in the queue of a node k if and only if node i is a descendant of node k and i has not been passed (see below) to the parent(s) of k. When the algorithm terminates, every queue will be empty.

3. An auxiliary integer vector of length n. Initially $vaux[]$ is all 0's.

The algorithm uses two basic operations. These operations are:

1. Pass(v,u). This operation takes the head of the queue of u, say node i, and if $v[i]$ is false, then it makes $v[i]$ true and appends i to the end of the queue of v.
2. Pop(v). This operation deletes the head of the queue of v.

In addition the algorithm uses the extra auxiliary integer array for each node v to avoid adding to the queue of v the same node number twice during one iteration. When two or more children of v have the same integer i at the head of the queue, only the one which succeeds writing to the location $vaux[i]$ during the parallel write, we do not care which one, will add i to the queue of v. The pass operations use fetch-and-adds to add to the tails of the queues. The maximum amount of delay caused by the parallel fetch-and-adds is $O(logn)$, because we have at most n children. Hence this algorithm can be implemented on the RP3 to run in $O(nlogn)$ time with $O(max(n, e))$ processors.

We want to prove the following theorem about the algorithm.

Theorem 8.1. *Let v_1 have v_h as a descendant. Then v_1 will pass to its parent(s) v_h within $2d$ iterations, where d is the number of descendants of v_1.*

Proof: Let $v_1, v_2, \ldots, v_h$ be a directed pointer path from v_1 to v_h. If none of the nodes v_i, $i < h$ on the path has another child, then v_h will be passed by v_1 in exactly h steps. Now suppose that some v_i has another child. Then that child may pass new descendants to v_i. Each new descendant that is passed to v_i is inserted into the sequence of node numbers $v_2, v_3, \ldots, v_h$ that is traveling towards v_1. Suppose that a new node k is inserted into the sequence. If it is inserted before v_h, then v_h will be delayed once. Now suppose that some other node on the path say v_j with $j < i$ also receives k. Then when k traveling towards v_1 reaches v_j, k will be eliminated from the sequence traveling upwards from v_j. However, since v_j was delayed once already by sending k towards v_1, the total number of delays caused by k is two. If some other node $v_l, l < j$ receives k, then it does not cause further delays. It could cause a second delay for the nodes between v_l and v_j, but will not cause a third delay for the nodes between v_j and v_i.

Similarly each new node can cause at most a delay of two iterations in v_h reaching v_1. If a node on the path is inserted into the sequence repeatedly, it will cause only one additional delay. Thus the total traveling time will be at most twice the number of descendants of v_1.

note: For directed acyclic graphs the two types of delays, one to pass through an extra node and second to pass an extra node turn out to overlap. Indeed, for directed acyclic graphs we can prove that if a vertex v has d descendants then d iterations are enough for v to pass its descendants upwards.

9 SINGLE SOURCE TRANSITIVE CLOSURE

In this section we return to the one root case, which is known as the single source transitive closure problem. The following combines the previous algorithm with an idea from [22]. To achieve a greater speed (and to keep our processor scheduling simple) we made two important changes. First, the algorithm (see Figure 5) has only one queue, where each entry is a pair (v, u) denoting a need to pass to v node number u, and in each iteration the whole queue is passed upward. Second, this pass operation is modified so that for each queue entry (v, u) if v is a descendant of the root, then u is passed directly to the root. Here we also keep for each node a list of parents.

Theorem 9.1. *The single source transitive closure algorithm finds the descendants of a given node in $O(\sqrt{n}logn)$ time with $O(\sqrt{n}max(n, e))$ processors.*

Proof: We at first show that $O(\sqrt{n})$ iterations are sufficient. Clearly, if node v has node u marked as a descendant, then there must be a path from v to u, because the algorithm always uses in marking only either passing from child to parent or transitivity.

Since there are n nodes, the maximum distance of a shortest path from the root to any node is $n - 1$. Take any node v with shortest distance l from the root, where $l < n$. Number the nodes on the path from the root to v: $1, 2, \cdots, l$. We show by induction that during the k^{th} iteration out of the nodes on the chosen path k new consecutive ones will be added to the descendants of the root. This holds initially. Then consider among all those nodes that are added to the descendants of the root during the k^{th} iteration *the maximum*, called max_k and the chain of nodes $max_k + 1, \ldots, max_k + k$. By our assumption, none of

```
For every vertex v in parallel do
    Set v[] to false.
    Set v[v] to true.
    Set vaux[] to zero.
    Set the queue to have (v,v) for each vertex v.
Repeat $\sqrt{2n}$ times
        For every queue entry (v,u) in parallel do
            write u to vaux[u] of u.
        For every queue entry (v,u) in parallel do
            If vaux[u] of v equals u then
                If false v[u] then
                    set v[u] to true.
                    For every parent $p$ of $v$ in parallel do
                        add entry (p,u)
            delete entry (v,u).
        For every queue entry (v,u) in parallel do
            If root[v] is true then
                change (v,u) to (root,u) in queue
Until
```

Figure 5: Single source transitive closure

the nodes on the chain are known descendants of the root. During the next iteration all of these k nodes will be added to the descendants of the root directly because each of them is a k^{th} descendant of some known descendant of the root. Hence, we have $max_{k+1} = max_k + k$. Therefore, $max_{\sqrt{2n}+1} = \sum_{i=1}^{\sqrt{2n}} i = n + \frac{\sqrt{2n}}{2}$ and after $O(\sqrt{n})$ time all nodes are marked, even the furthest away.

To analyze the processor requirement we use Brent's theorem [5] that *any synchronous parallel algorithm taking time t that consists of a total of x elementary operations can be implemented by p processors within a time of* $\frac{x}{p} + t$. Clearly, our algorithm takes $O(\sqrt{n} logn)$ time with $O(ne)$ processors because in each iteration in parallel through each edge we pass at most n nodes. Moreover, since passes are distinct the total work done is also $O(ne)$, hence by Brent's theorem it can be done by $O(\sqrt{n} max(n, e))$ processors in $O(\sqrt{n} logn)$ time, which proves the theorem.

note: The algorithm can be extended to do "breadth-first search", i.e. to find the length of the shortest path to each node because we reach each node on the shortest path. The depth of a node reached first in the k^{th} iteration is k plus the minimum depth of the nodes reporting it as a known descendant in their queues. The unit weight shortest path to each node from the root can be found by associating the newly reached node r with the minimum-depth ancestor q and putting them as a pair into the root's vector. We also associate in one-depth passes each node with its parent by passing the edges (p, c) for parent p and child c upward while treating them as we did the children. Then the shortest path from r to q can be read from q's memory. These modifications do not increase significantly the time, processor, or space bounds.

10 CONCLUSIONS

Several algorithms and parameters, such as c, which in turn depends on p and q, need an average case analysis, although a convincing one seems complicated and elusive. However, we hope to do further testing on the completed RP3 with real list processing systems.

Acknowledgements: I would like to thank Alan Norton, Whei-Ling Chang and other members of the RP3 research group for their encouragements and the excellent environment provided for conducting this research.

References

[1] Aho, A. V., Hopcroft, J. E., and Ullman, J. D. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] Awerbuch, B., A. Israeli, and Y. Shiloach, "Finding Euler Tours in Logarithmic Parallel Time", Proc. of the 16th ACM Symposium on Theory of Computing, (1984), 249-254.

[3] Awerbuch, B. and Y. Shiloach, "New Connectivity and MSF Algorithms for Ultracomputer and PRAM", Proc. of the 1983 International Conference on Parallel Processing, IEEE Computer Society Press, 175-179.

[4] Ben-Ari, Mordechai "Algorithms for On-the-Fly Garbage Collection" ACM Trans. Prog. Lang. Sys. 6,3 (July 1984), 333-345.

[5] Brent, R. P., "The Parallel Evaluation of General Arithmetic Expressions", J. ACM, 21:2, 1974, 201-208.

[6] Chin, F.Y.,J. Lam, and I. Chen."Optimal Parallel Algorithms for the Connected Component Problems." Proc. 1981 International Conference on Parallel Processing, IEEE Computer Society Press, 170-175.

[7] Cohen, J. "Garbage Collection of Linked Data Structures." Comput. Surv. 13,3 (Sept. 1981), 341-367.

[8] Cole, R. and U. Vishkin, "Deterministic Coin Tossing and Accelerating Cascades: Micro and Macro Techniques for Designing Parallel Algorithms." 1985.

[9] Dijkstra, E. W., L. Lamport, A. J. Martin, C. S. Scholten, and E. F. Steffens "On-the-Fly Garbage Collection: an Exercise in Cooperation." Commun. ACM 21,11 (Nov. 1978), 966-975.

[10] Gazit, Hillel, "An Optimal Randomized Algorithm for Finding Connected Components in a Graph", IEEE FOCS, 1986, 492-501.

[11] Gottlieb, A., and J. T. Schwartz "Networks and Algorithms for Very Large Scale Parallel Computations", Computer 15,1 (1982), 27-36.

[12] Hirschberg, D. S., A. K. Chandra, and D. V. Sarwate, "Computing connected Components on Parallel Computers", Commun. ACM 22,8 (1979), 461-464.

[13] Hickey, Tim and J. Cohen, "Performance Analysis of On-the- Fly Garbage Collection", Commun. ACM 27,11 (Nov. 1984), 1143-1154.

[14] Kieburtz, R. B., "The G-machine: a Fast, Graph-reduction Evaluator", Proc. of IFIP Conf. on Functional Programming Languages and Computer Arch., Nancy, 1985, 400-413.

[15] Kung, H. T. and S. W. Song "An Efficient Parallel Garbage Collection System and its Correctness Proof." IEEE FOCS, 1977, 120-131.

[16] Klein, P. N. and J. Reif "An Efficient Parallel Algorithm for Planarity", Proc. of the 27th IEEE Symp on the Foundations of Computer Science (1986), 465-477.

[17] Miller, G. L. and V. Ramachandran "A New Graph Triconnectivity Algorithm and its Parallelization", Proc. of l9th ACM Symp. on Theory of Computation, (1987), 335-344.

[18] Pfister, G.F., W.C. Brantley, D.A. George, S.L. Harvey, W.J. Kleinfelder, K.P. McAuliffe, E.A. Melton, V.A. Norton, and J. Weiss "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture" IEEE Computer Society Press, 764-771.

[19] Reif, J. H. "Optimal Parallel Algorithms for Graph Connectivity", Center for Computing Research TR-08-84, Harvard University.

[20] Revesz, G., *Lambda-Calculus, Combinators, and Functional Programming*, Cambridge Univ. Press, 1988.

[21] Revesz, P. "A New Parallel Garbage Collection Algorithm", B.S. thesis, Tulane Univ., 1985.

[22] Revesz, P. "Graph Reachability Problems", Brown Univ. Tech. Report CS-87-02.

[23] Revesz, P. "Parallel Garbage Collectors for the RP3", IBM T.J. Watson Report RC-13130, 1987.

[24] Shiloach, Y. and U. Vishkin, "An O(log n) Parallel Connectivity Algorithm" Journal of Algorithms 3, 57-67 (1982).

[25] Steele, G. L. Jr. "Multiprocessing Compactifying Garbage Collection", Commun. ACM 18,9 (Sept. 1975), 495-508.

[26] Stone, J., "A Parallel Lambda-calculus Interpreter", IBM T.J. Watson Report, Jan. 1987.

[27] Tarjan, R.E. and U. Vishkin, "An Efficient Parallel Biconnectivity Algorithm", SIAM J. of Computing, 14,4 (1985), 862-874.

PARALLEL ALGORITHMS FOR SCHEDULING TASKS WITH PRECEDENCE CONSTRAINTS ON A SINGLE MACHINE

ELIEZER DEKEL
Computer Science Program
The University of Texas at Dallas
Richardson, TX 75083-0688
U.S.A.

CHII-HUAH SHYU
Advanced Technology Center
Electronics Research & Service Organization
Industrial Technology Research Institute
Taiwan, R. O. C.

Abstract: Given are n tasks that are to be processed by a single machine (one at a time, with no interruptions), subject to arbitrary precedence constraints. Associated with each task T_j is a known processing time p_j and a monotone nondecreasing cost function $c_j(t)$, giving the cost that is incurred by the completion of the task T_j at time t. The problem is to find a schedule which will minimize the maximum of the incurred cost. We present several parallel algorithms for this problem on the exclusive-read exclusive-write parallel random access machine.

1. Introduction

Machine scheduling problems with various objective functions have been extensively studied, leading to many efficient algorithms. As a well investigated area, problems from scheduling theory for which efficient polynomial time sequential algorithms exist, where used in order to develop and study parallel algorithmic techniques. For example Dekel and Sahni [4] presented an O($\log^2$ n) time, using O(n^2) PEs parallel algorithm to construct a schedule with the fewest tardy tasks. They also showed that the optimal single machine scheduling with earliness and tardiness penalties can be solved in time O(log n) using O(n^2) PEs. Using the recent sorting algorithm of Cole [1], this problem can be solved in O(log n) time with only O(n) PEs.

Scheduling problems were also studied in the context of complexity classes *NC* , *RNC* and *P*-complete. It is generally believed that problems in the class *P*-complete (polynomial time complete under log-space reduction) are *inherently sequential*. For problems in the class *NC* (solvable in poly-logarithmic time using a polynomial number of processors), one can effectively utilize more then one processor in their solutions. Recently Helmbold and Mayr [8] presented a poly-logarithmic deterministic solution for the two processor scheduling problem. The class *RNC* consists of problems that are solvable using a polynomial number of processors in poly-logarithmic time with high probability. Parallel scheduling algorithms for these and other problems can be found in [4-8,11].

In this paper we study the following single machine scheduling problem [10]: Suppose n tasks are each to be processed by a single machine (one at a time, with no interruptions), subject to arbitrary precedence constraints. Associated with each task T_j is a known processing time p_j and a monotone nondecreasing cost function $c_j(t)$, giving the cost that is incurred by the completion of the task T_j at time t. The problem is to find a sequence which will minimize the maximum of the incurred cost.

Lawler showed [10] that this problem can be solved using the following rule: From among all tasks that are eligible to be sequenced last; i.e., that have no successors under the given precedence constraints, put the task which will incur the smallest cost in that position. Then repeat on the remaining n-1 tasks, and so on. The algorithm can be implemented in $O(n^2)$ time, under the assumption that the cost functions $c_j(t)$, $1 \leq j \leq n$ can be computed in unit time for any value of t.

The model of computation used here is the *parallel random access machine* (P-RAM) in which all processing elements (PEs) have access to a common memory and run synchronously [12]. Our presentation focuses on the *exclusive read exclusive write* (EREW) P-RAM. An EREW P-RAM does not allow simultaneous reading from or writing to the same memory location.

This paper is organized as follows. In section 2, we define the decision problem associated with this problem. We show that the decision problem is in *NC* by presenting a poly-logarithmic time parallel algorithm for solving it. We develop a parallel algorithm for the general problem in section 3. This algorithm runs in poly-logarithmic time but uses pseudo-polynomial number of processors. In section 4 we present a Las Vegas algorithm for the general problem. Theorem proofs are omitted in this version of the paper and can be found in [6]. Concluding remarks and suggested open problems are given in the final section.

2. The Decision Problem

The following assumptions are made in this paper:

1. The processing time of each task is an integer. While the serial solution will work for processing times that are real numbers it is critical for the parallel solution to have integer processing times.

2. Given cost B, the largest integer value x such that $c_i(x) \leq$ B can be computed in constant time for each task. This is a natural assumption given that in the original formulation of the problem [10] it is assumed that $c_i(t)$ can be computed in constant time for any t.
3. The task with the smallest processing time requires at least 1 unit of processing time. This assumption follows naturally from assumption 1.

Instead of solving the required scheduling problem directly, let's consider the following *decision problem*:

Instance: Given an n tasks system with arbitrary precedence constraints and a cost B. Associated with each task i, $1 \leq i \leq n$ is a processing time $p_i \in Z^+$ and a monotone nondecreasing cost function $c_i(t)$.

Question: Is there a nonpreemptive feasible schedule of these tasks on a single machine so that the maximum incurred cost is B or less?

Some notations: If A is a set, the $|A|$ is the number of elements in A. We use f to denote the total processing of the n tasks, i.e., $f = \sum_{k=1}^{n} p_k$. The precedence constraints are given as a relation on the set of tasks. If task T_i precedes task T_j (denoted $T_i \rightarrow T_j$) then task T_i must be completed before task T_j can be started. $T_i * \rightarrow T_j$ denotes that there is a chain of tasks $T_i, T_{i1}, \ldots, T_{ik}, T_j$ such that $T_i \rightarrow T_{i1} \rightarrow \cdots \rightarrow T_{ik} \rightarrow T_j$. Task T_i is a *predecessor* of task T_j and T_j is a *successor* of T_i. All the *predecessors* of task T_j must be completed before it can be started. We claim that a task system is schedulable under the cost bound B if and only if there exists a feasible sequence of tasks (that is no task is violating any of the precedence constraints) such that the maximum incurred cost is no greater than B.

Procedure CheckBound (Figure 1.) solves the decision problem. It works as follows: In lines 1-3, we find r' (i), the latest time in which task T_i can terminate incurring a cost of B or less. Observe that the machine is never idle in the interval [0,f]. In lines 4-5 we refine the value of the latest time in which a task can terminate without violating the B cost bound. r(i) is an upper bound for the termination time of task T_i when the precedence constraints are taken into account. If there are two tasks $T_i * \rightarrow T_j$ and r' (j) < r' (i), then the completion of task T_i at r' (i) will force task T_j be finished later than r' (i). Hence the incurred cost will exceed B even though task T_i is finished at r' (i). Given the calculated

```
line procedure CheckBound(B)
1    in parallel, for all $T_i$, $1 \le i \le n$
2        $r'(i) \leftarrow \min\{ f, \max\{ x \mid c_i(x) \le B\}\}$
3        $suc(i) \leftarrow \{T_j \mid T_i * \rightarrow T_j\} \cup T_i\}$
4    in parallel, for all $T_i$, $1 \le i \le n$
5        $r(i) \leftarrow \min\{r'(j) \mid T_j \in suc(i)\}$
6    in parallel, for all $T_i$, $1 \le i \le n$
7        $RANK(i) \leftarrow | c1(i) | + | c2(i) | + | c3(i) |$
8    in parallel, for all $T_i$, $1 \le i \le n$
9        if $RANK(i) = O$ then $s(i) \leftarrow O$
10       else $s(i) \leftarrow \sum_{RANK(k) < RANK(i)} p_k$
11       if $c(s(i) + p_i) > B$ then mark unschedulable
12   end
```

Figure 1. procedure CheckBound.

upper bound we now compute the rank of each task (lines 6-7). We define a rank for each task such that if RANK(i) < RANK(j) then task T_i should be scheduled before task T_j. The set of tasks scheduled before task T_i, $1 \le i \le n$, is composed of the following three components:

1. $c1(i) = \{ T_j \mid r(j) < r(i) \}$.
2. $c2(i) = \{ T_j \mid r(j) = r(i) \text{ and } | suc(j) | > | suc(i) | \}$.
3 $c3(i) = \{ T_j \mid r(j) = r(i), | suc(j) | = | suc(i) | \text{ and } j < i \}$.

The validity of this step is discussed and proved in lemma 1 and theorem 1 below. In lines 8-10 we compute s(i) the starting time of task T_i. The computation of the starting time is based on the rank computed in line 7. Finally in line 11 we check for each task if its incurred cost does not violate the set bound. We can modify procedure CheckBound so that it produces the schedule for the task system if we add the following line:

11′ else schedule task T_i to be processed in interval $(s(i), s(i) + p_i]$.

Lemma 1. If RANK(i) < RANK(j), then task T_i would not be a successor of task T_j.

Theorem 1. Let $\beta = \{ i_1, i_2, \ldots, i_n \}$ be a permutation of the n tasks of our decision problem, such that RANK(i_1) $\leq$ RANK(i_2) $\leq \cdots \leq$ RANK(i_n). The task system is schedulable under the cost bound B if and only if it can be processed in order β so that the maximum incurred cost is no greater than B.

Theorem 2. The decision problem can be solved in $O(log^2 n)$ time using O(n^3/log n) PE2 on an EREW P-RAM.

3. The General Parallel Algorithm

There are n possible costs at each point t, $1 \leq t \leq f$. In the worst case, there are O(nf) points as the candidates of the lowest cost bound. We can solve the general problem by first sorting these O(nf) points and then use the binary search method together with procedure CheckBound (Figure 1.) to find the lowest cost bound B such that the maximum incurred cost of these n tasks is B or less. The sorting step takes O(log nf) time using O(nf) PEs [1]. The transitive closure of the precedence relation has to be computed only once. Procedure CheckBound will be called O(log nf) times. Hence this method takes O((log nf)(log n)+log^2 n) time using O(nf+n^3/log n) PEs.

Another method for solving the general problem is to perform a binary search on the possible costs on each time point in parallel. At each time point, there are n possible costs, so each binary search will iterate O(log n) times. The transitive closure of the precedence relation is computed once in O(log^2 n) time using O(n^3/log n). On each iteration of the binary search, procedure CheckBound is called. Since the transitive closure is available procedure CheckBound can perform its computation in O(log n) time using O(n^2/log n) PEs. Once the lowest cost bound on each time point is found, it takes another O(log f) time with o(f) PEs to find the minimum of these cost bounds. The overall complexity of this method is therefore, O(log^2 n + log f) time using O(n^3/log n + fn^2/ log n) PEs. The f concurrent invocations of procedure CheckBound bring up the possibility of read conflicts. These can be avoided if f copies of the data are made. The data replication can be performed in O(log f) time using o(fn^2/log n) PEs.

Both methods described above run in pseudo-logarithmic time and require a pseudo-polynomial number of PEs and therefore are impractical for instances of the problem when f is big. In the rest of this section we present a more efficient approach for solving the general problem. While this method still runs in pseudo=logarithmic time and requires a pseudo-polynomial number of PEs to perform, its development provides us with the tools for our randomized algorithm in the next section. The idea is that we find the necessary cost bound at each point t, $1 \le t \le f$. Then, the lowest cost bound is the maximum of the cost bounds obtained at points t, $1 \le t \le f$.

```
line  procedure DGEN
1     P ← min { $p_i$ | $1 \le i \le n$ }
2     X ← min { $c_i$(f) | $1 \le i \le n$ }
3     Y ← max { $c_i$(P) | $1 \le i \le n$ }
4     B ← max {X, Y}
5     in parallel, for all $t$, $P \le t \le f$
6             in parallel, for all $T_i$, $1 \le i \le n$
7                     s ← max { $c_j$ (t) | $T_i \ast\!\to T_j$ }
8                     $c'_i(t)$ ← max {$s$, $c_i(t)$}
9     SDGEN(P, B, NS)
10    in parallel, for all $t \in NS$
11            LB(t) = min { $c'_i$ (t) | $\sum_{c'_k(t) \le c'_i(t)} p_k \ge (f - t + 1)$ }
12    B ← max { LB(t) | t ∈ NS }
13    CheckBound(B)
14    end
```

Figure 2. Procedure DGEN

```
line  procedure SDGEN(P, B, NS)
1     NS ← ∅
2     in parallel, for all $t$, $P \le t \le f$
3             GAP(t) ← ( f - t + 1) - $\sum_{c'_k(t) \le B} p_k$
4     if GAP(t) > 0 then NS ← NS ∪ {t}
5     end
```

Figure 3. Procedure SDGEN

The main procedure is DGEN. Lines 1-4 initialize B with the lowest cost bound. For any two tasks T_i and T_j such that $T_i \rightarrow T_j$ and $c_i(t) < c_j(t)$. If task T_i is finished at t, then task T_j will be forced to be finished later than t, which makes the incurred cost for such a schedule at least c_j (t). Based on this observation, lines 5-8 of DGEN modify the cost functions for all tasks in the range [P,f]. In line 9 procedure SDGEN (Figure 3.) is called to collect the points where the necessary and sufficient conditions for the task system to be schedulable do not hold. The correctness of SDGEN and lines 10-12 in which we compute the lowest cost bound is given in theorem 3 below. In line 13 we use the minimal maximum incurred cost for the given task system (computed in line 12), as a parameter for the modified procedure CheckBound (with line 11′). This procedure constructs the optimal schedule.

Theorem 3. GAP(t) = (f - t + 1) - $\sum_{c_k(t) \le B} P_k \le 0$ for $1 \le t \le f$ are a necessary and sufficient condition for the task system to be schedulable under cost bound B.

Theorem 4. The general problem can be solved in O(log f + $\log^2$ n) time using O(fn^2/log n + n^3/log n) PEs.

4. The Randomized Parallel Algorithm

In this section we present a Las Vegas algorithm to solve the general problem. The term "Las Vegas algorithm" was coined by L. Babai to characterize those randomized algorithms that do not lie: They either give the correct answer or (with low probability) announce that they have failed to find the answer [9].

We define a test point to be a point t, $1 \le t \le f$, where GAP(t) > 0. Observe that if the cost bound is raised to LB(t_0) at t = t_0 and $\sum_{c'_k(t_0) \le LB(t_0)} p_k - (f - t_0 + 1) = k \ge 0$, then there is no test point in the interval $(t_0 - k - 1, t_0]$.

Let $E[x]$ be the expected processing time of a task in the n task system. For the analysis of the performance of the randomized parallel algorithm, we define the following functions:

UB(t) = max { $c'_i(t) \mid \sum_{c'_k(t) \le c'_i(t)} p_k < (f - t + 1)$ }

$$LB(t) = \min \{ c'_i(t) | \sum_{c'_k(t) \le c'_i(t)} p_k \ge (f - t + 1) \}$$

$$TUB(t) = \sum_{c'_k(t) \le UB(t)} p_k$$

$$TLB(t) = \sum_{c'_k(t) \le LB(t)} p_k$$

We Assume that E[TLB(t)] + E[TUB(t)] ≈ 2(f - t + 1). Since E[TLB(t)] - E[TUB(t)] ≥ E[x], we have E[TLB(t)] - (f - t + 1) = (E[TLB(t)] - E[TUB(t)])/2 ≥ E[x]/2.

In the worst case, there are $O(f) = O(n E[x])$ test points. If n test points are chosen to raise cost bound, then there are intervals of total expected length $O(n E[x]/2)$ that do not contain test points. In the first step of our randomized parallel algorithm we select a lower bound for the minimal maximal incurred cost. This is done the same way as lines 1-4 of procedure DGEN. Let the intervals containing test points be in set G. We start by assuming that $G = [P, f]$ interval with test points. We select n equally spaced points from this interval as test points and compute the value of the GAP function for each. Next we comput LB() for each of the points. For each new LB() value, we compute the resulting interval with no test points. This computed intervals are then removed from G. At this point we compute the maximum of the computed LB() values. This is now the candidate for a lower bound on the maximum incurred costs. Next we use CheckBound with the new lower bound as a parameter. If Checkbound can schedule the task system with the supplied bound then we terminate. Otherwise, we find again n equally spaced points in G and repeat the process.

It can be easily seen that the randomized algorithm always produces the lowest cost bound such that the task system is schedulable. All the steps of this algorithm were analyzed and explained in the context of our previous algorithms. For example we show in Theorem 4, LB() can be computed in time $O(\log n)$ using $O(n^2)$ PEs if there are n points chosen. Since the randomized parallel algorithm will be terminated after few iterations. Clearly its time complexity $O(\log^2 n)$ using $O(n^3/\log n)$ PEs.

5. Concluding Remarks

We have developed several parallel algorithms for the single machine scheduling problem and have shown the following interesting properties:

(1) The decision problem can be solved in O($\log^2$ n) time using O(n^3/log n) PEs.

(2) When all tasks have unit processing time, then the schedule with minimum maximum incurred cost can be constructed in O($\log^2$ n) time using O(n^3/log n) PEs.

(3) If the assumption that the cost functions can be computed in unit time is removed; i.e., $c_i(t)$, $1 \leq t \leq f$, must be specified for each task T_i, $1 \leq i \leq n$, in the most general case, then this problem is in NC^2.

(4) The problem is in RNC^2 under the assumption that E[TLB(t)] + E[TUB(t)] ≈ 2(f - t + 1), $1 \leq t \leq f$.

(5) If the transitive closure of the precedence relation is known or a more efficient transitive closure algorithm becomes available then the problem is in *RNC*

It remains open whether the general problem is in *NC* or whether the problem with processing times drown from R has a parallel solution.

References

[1] R. Cole, "Parallel merge sort," Proc. 27th IEEE Symp. on Foundation of Computer Science, (1986) 511-517.

[2] S. A. Cook, "Toward a complexity theory of synchronous parallel Computation," L'Enseigment Mathematique XXVII, (1981) 99-124.

[3] E. Dekel, D. Nassimi and S. Sahni, "Parallel matrix and graph algorithms," **SIAM J. Comput.** (1981) 657-675.

[4] E. Dekel and S. Sahni, "Parallel scheduling algorithms," **Operations Research**, 31 (1983) 24-49.

[5] E. Dekel and S. Sahni, "Binary trees and parallel scheduling algorithms," **IEEE Trans. Comput.**, C-32 (1983) 307-315.

[6] E. Dekel and C-H Shyu, "Parallel algorithms for single machine scheduling subject to precedence constraints," U. T. D. Computer Science Technical Report 1989.

[7] D. Dolev, E. Upfal and M. Warmuth, "Scheduling tree in parallel," Proc. VLSI: Algorithms and Architectures, International Workshop on Parallel Computing and VLSI, Italy, 1984.

[8] D. Helmbold and E. Mayr "Two processor scheduling is in *NC* ," **SIAM J. Comput.** 16 (1987) 747-759.

[9] D. S. Johnson, "NP-completeness column," **J. Algorithms**, (1984) 433-447.

[10] E. L. Lawler, "Optimal sequencing of a machine subject to precedence constraints," **Management Science**, (1973) 544-546.

[11] U. V. Vazirani and V.V. Vazirani, "The two processor scheduling problem is in *RNC* ," Proc. 17th ACM Symp. on Theory of Computing, (1985) 11-21.

[12] U. Vishkin, "Synchronous parallel computation - a survey," TR 71, Dept. Computer Science, Courant Inst., NYU (1983).

PARALLEL RANDOM BIT GENERATION ON SIMD ARCHITECTURES

MARTIN L. BRADY
Lockheed R&D Div.
3251 Hanover St.
Palo Alto, CA 94304

We consider the problem of generating pseudo-random bit (PRB) sequences in parallel on SIMD (Single Instruction–Multiple Data Stream) architectures. The goal is that each processor produce a new random bit in one parallel pass of the PRB generation algorithm, i.e., the generator should ideally model an independent coin toss at each of the processors. Such an algorithm has many practical applications, e.g., in simulation and modelling of physical systems (see, e.g., [TM87]). This problem is quite different from that of sequential generation of PR numbers, in which a single sequence of n-bit uniformly distributed integers is produced.

We consider a parallel SIMD model in which all P processors are simple bit-serial elements, which execute the same instruction synchronously. We assume that processors are connected in a 4-nearest-neighbor mesh with wraparound (but our algorithms are easily extended to other interconnection patterns). Each processing element has a limited amount of local memory.

The PRB generator will typically be used to generate random bits used by some larger randomized algorithm. We must take care to limit the time and space used for PRB generation so that the host algorithm is not greatly affected. We therefore define the following constraints for a SIMD algorithm for PRB generation.

1. Each processor produces a random bit in parallel in a small constant number of cycles, independent of the number of processors.
2. The algorithm uses only a small constant number of bits of memory per processor.

A well known method for generating a single PRB sequence is by a *linear feedback shift register (LFSR)*, in which a single random bit is generated from the previous n bits, by the relation $x_t = \sum_{i=1}^{n} c_i x_{t-i} \pmod 2$, where $c_i=1$ represents a feedback connection from the i^{th} delay unit [T65], [G67]. Pearson has shown how to interconnect individual shift registers so that the period of the PRB sequences are determined by the *sum* of the storage of all of the shift registers, i.e., so that they act as one long LFSR with multiple feedback points [P83]. We further generalize this method to allow feedback between any pair of bits of the generator, which we refer to as a *linear feedback array (LFA)*. We then show how to choose interconnections which satisfy our requirements. The maximum period of such a generator is 2^n-1, and sequences of maximum length are known to have desirable randomness properties (see, e.g., [G67]). A period of 2^n-1 is guaranteed if the following hold.

1. 2^n-1 is prime (called a Mersenne prime).
2. The characteristic polynomial is irreducible.

The *characteristic polynomial*, $f(\lambda)$, depends only upon the feedback connections. For a LFSR, $f(\lambda) = 1 + \sum_{i=1}^{n} c_i\lambda^i$. The characteristic polynomial for the LFA is more complex, $f(\lambda) = \det(\mathbf{I}+\mathbf{D})$, where element d_{ij} of $\mathbf{D}$ represents the feedback connection from processor i to processor j. For example, $d_{ij} = \lambda^2+\lambda^3$ implies that j receives feedback from i delayed by 2 and 3.

This work was supported by internal Lockheed funds under IR&D project RDD504.

Our strategy for designing a parallel PRB generation algorithm is as follows.

1. *Select a Mersenne prime, $2^n - 1$, such that $n>P$.*

   Since the number of shift register bits per processor is $\lceil n/P \rceil$, n should be no more than a small constant times P.

2. *Select a set of feedback connections, such that the order of $f(\lambda)$ is n.*
   In order to keep the running time small, feedback connections are restricted to nearest neighbors, in addition to self interconnections. Further, the number of different connection *types* (e.g., north neighbor, delay 3) is limited to a small constant, since each type takes additional time, and requires additional storage to mask the processors which employ the connection.

3. *Calculate $f(\lambda) = \det(\mathbf{I}+\mathbf{D})$.*

4. *Determine whether $f(\lambda)$ is primitive.*

We repeat 2 through 4 until a suitable set of interconnections is found. Unfortunately, both calculations 3 and 4 can be quite time consuming for large n. Restricting the connections to neighbors limits the number of non-zero elements in **D**, but the determinant still may have a large number of terms. However, we have found a more restricted class of connections which give simple determinant calculations, but still allow enough freedom to choose an irreducible characteristic polynomial. Each processor has two feedback connections: one to its west neighbor, and one either to itself or its north neighbor. Specifically, in each row of the mesh, exactly one processor has a north connection, the rest have self connections. There are $(\sqrt{P})^{\sqrt{P}}$ ways to place these $\sqrt{P}$ north connections in a square mesh. This arrangement gives a determinant with only three terms, so $f(\lambda)$ is easily calculated.

A typical sequential algorithm for determining whether a polynomial of order n is primitive runs in $O(n^2k)$ time, where k is the number of terms in the polynomial (k is commonly $\Theta(n)$ in our application). However, we can use the target mesh architecture itself to perform the irreducibility test. We have designed a parallel algorithm which which runs in time $O(n^2)$ on an n processor mesh architecture.

We are currently using the method presented to implement a PRB generator on a 10,000 processor mesh architecture, using $n = 44497$. This will allow us to evaluate the algorithm and check the correlations between different bit sequences.

When P is extremely large, it may difficult to find a suitable primitive characteristic polynomial for the generator. The time to test for primitivity can become prohibitive even in parallel, and furthermore, there may not be a known Mersenne prime in the proper range. We are currently investigating ways to trade off the period of the sequence for the time to construct a generator. Of course, one could implement different independent generators on, say, each row of the mesh, using different characteristic polynomials but with similar sets of connections. However, this would require discovering $\sqrt{P}$ different primitive polynomials, having a similar set of interconnection types.

References

[G67] Golumb, S.W., *Shift Register Sequences*, Holden-Day, San Francisco, 1967.

[P83] Pearson, R. B., "An Algorithm for Pseudo Random Number Generation Suitable for Large Scale Integration", *J. Comp. Phys.* **49**, pp. 478-489, 1983.

[T65] Tausworth, R. C., "Random Numbers Generated by Linear Recurrence Modulo Two", *Math. Comput.* **19** (90), pp. 201-209, 1965.

[TM87] Toffoli, T., and N. Margolis, *Cellular Automata Machines*, MIT Press, Cambridge, MA, 1987.

BULK–SERVICE QUEUE WITH SERVICE DELAY POLICY

RAYMOND W. YEUNG

AT&T Bell Laboratories

Holmdel, NJ 07733

Abstract – A Poisson stream of passengers arrive at a bus terminal. The bus in service can take B passengers per ride, and the passengers are served on an FCFS basis. The passengers (at most B of them) will board the bus as soon as it arrives. If the bus is already full, it will depart immediately, otherwise the bus driver will decide to depart immediately or at a later time according to a policy $\mathcal{P}$. Then the bus will deliver the passengers to a destination and return to the terminal. The time it takes for the round trip is described by a probability distribution which is independent of the number of passengers on the bus. For a general class of policies, which includes policies with timers, we show that this seemingly complicated problem can be decomposed into two tractable subproblems which can be treated independently. We discover that the stationary behavior of the queueing system depends only on the *conditional departure distribution* of the policy. Exact solution can be obtained if the round trip time distribution is Erlangian.

1. INTRODUCTION

In this paper we study a bulk–service queueing system with service delay policy. The results obtained are general and are applicable to situations that frequently arise in many real systems.

The queueing system we consider is described as a transportation situation as follows. Passengers arrive at a bus terminal as a Poisson stream of intensity λ and form a queue. The bus can take B passengers per ride, and the passengers are served on an FCFS basis. The passengers (at most B of them) will board the bus as soon as it arrives. If the bus is already full, it will depart immediately, otherwise the bus driver will decide to depart immediately or at a later time according to a policy $\mathcal{P}$. The passengers who arrive at the terminal while the bus is waiting will board the bus immediately. It is required that $\mathcal{P}$ satisfies the following assumptions:

(i) the probability that the bus will wait for a certain time at the terminal is a function of the number of passengers in the queue when the bus arrives and the arrivals of the passengers during the time the bus is waiting;

(ii) the bus departs immediately once it is full.

We will refer to this class of policies as $\mathcal{C}$, which is general enough to include most legitimate policies (e.g., policies with timers). Assumption (ii) ensures that the bus does not wait unnecessarily at the terminal. Then the bus will deliver the passengers to a destination and return. The time it takes for the round trip is described by a probability distribution g(t) with mean $1/\mu$, which is independent of the number of passengers on the bus. The exact time the passengers arrive at the destination is not of concern. We are interested in the stationary behavior of the queueing system.

In this paper we present a universal approach to this problem. We show that for any policy in $\mathcal{C}$, it is possible to decompose the problem into two tractable subproblems which can be treated independently. We discover that the stationary behavior of the system depends only on the *conditional departure distribution* of the policy. In Section 2 we will discuss an imbedded Markov chain which is invariant to the particular policy. The efficiency of the system, the stationary distribution of the number of passengers in the queue and the sojourn time distribution will be determined in Sections 3, 4 and 5, respectively.

2. IMBEDDED MARKOV CHAIN AT BUS ARRIVAL EPOCHS

It is clear that the system is ergodic if $\lambda < B\mu$. Let N_k be the number of passengers in the queue when the bus returns from the kth service. Then $\{N_k\}$ is an imbedded Markov chain, because N_k summarizes the past history of the process. Let A_k be the number of passenger arrivals during the kth service. If N_k is not less than B, then the bus will take B passengers, leaving (N_k-B) passengers at the terminal. When the bus returns from the (k+1)st service, there will be (N_k-B+A_{k+1}) passengers at the terminal. If N_k is less than B, no passenger will be left behind when the bus departs, and there will be A_{k+1} passengers when the bus returns. Therefore

$$N_{k+1} = (N_k-B)^+ + A_{k+1},$$

where

$$(y)^+ = \begin{cases} y & \text{if } y \geq 0 \\ 0 & \text{if } y < 0 \end{cases}.$$

Note that this imbedded Markov chain is universal to $\mathcal{C}$ (that is, it does not depend on the particular policy in $\mathcal{C}$). We adopt the notations that X(z) is the z–transform of the distribution of a discrete random variable X, and $F^*(s)$ is the Laplace Transform of the distribution f(t). When a sequence of random variables Y_k converges in law, we denote the limiting variable by Y. Similar notations will follow naturally. Note that $\{N_k\}$ is aperiodic and irreducible, and is positive recurrent if $\lambda < B\mu$. Then the probability distribution of N_k will converge to a stationary distribution $\{ p_j, j \geq 0 \}$, where

$$p_j = \lim_{k\to\infty} P[N_k=j] = P[N=j] ,$$

so

$$N(z) = \lim_{k\to\infty} N_k(z) = \sum_{j=0}^{\infty} p_j z^j .$$

It was obtained by Bailey [1] and other authors in less general context that

$$N(z) = \frac{A(z) \sum_{i=0}^{B-1} p_i (z^B - z^i)}{z^B - A(z)} .$$

This expression contains B unknowns, namely p_i , $0 \leq i \leq B-1$. Since these unknowns will not impair our discussion in the rest of the paper, we temporarily assume that they can be determined. It can be shown that they can be determined if g(t) is Erlangian [4], and can be determined explicitly if g(t) is exponential.

3. EFFICIENCY

Let D_k be the number of passengers on the bus when it departs for the kth service, and

$$q_{\mathcal{P}}(j|i) = P[D_{k+1}=j | N_k=i] , \qquad 0 \leq i \leq B-1 , i \leq j \leq B$$

be the *conditional departure distribution* of policy $\mathcal{P}$, which is readily determined from the specification of $\mathcal{P}$ and the arrival process. Note that for all i,

$$\sum_{j=i}^{B} q_{\mathcal{P}}(j|i) = 1 .$$

The significance of conditional departure distribution will be clear once it is proved that the stationary behavior of the process depends only on the conditional departure distribution of the policy.

The efficiency of the system η is defined as the expected number of passengers on the bus when it departs. We first determine $P[D=j]$, $0 \leq j \leq B$. (Note that we do not assume the bus cannot depart with no passenger aboard.) For $0 \leq j \leq B-1$,

$$\begin{aligned} P[D_{k+1}=j] &= \sum_{i=0}^{j} P[D_{k+1}=j | N_k=i] P[N_k=i] \\ &= \sum_{i=0}^{j} q_{\mathcal{P}}(j|i) P[N_k=i] , \end{aligned}$$

and for j = B ,

$$
\begin{aligned}
P[D_{k+1}=B] &= \sum_{i=0}^{B-1} P[D_{k+1}=B \mid N_k=i]P[N_k=i] + \sum_{i=B}^{\infty} P[N_k=i] \\
&= \sum_{i=0}^{B-1} q_{\mathcal{P}}(B \mid i)P[N_k=i] + \sum_{i=B}^{\infty} P[N_k=i] .
\end{aligned}
$$

Letting k → ∞ , we obtain

$$
P[D=j] = \begin{cases} \displaystyle\sum_{i=0}^{j} q_{\mathcal{P}}(j \mid i)p_i & 0 \le j \le B-1 \\ \displaystyle\sum_{i=0}^{B-1} q_{\mathcal{P}}(B \mid i)p_i + \sum_{i=B}^{\infty} p_i & j = B \end{cases} .
$$

Therefore

$$
\begin{aligned}
\eta &= \sum_{j=1}^{B} P[D=j]j \\
&= \left[\sum_{j=1}^{B-1} \sum_{i=0}^{j} (q_{\mathcal{P}}(j|i)p_i)\, j \right] + B \left[\sum_{i=0}^{B-1} q_{\mathcal{P}}(B \mid i)p_i + \sum_{i=B}^{\infty} p_i \right] \\
&= \sum_{j=0}^{B-1} \sum_{i=0}^{j} (q_{\mathcal{P}}(j|i)p_i)\, j + B \sum_{i=0}^{B-1} q_{\mathcal{P}}(B \mid i)p_i + B \sum_{i=B}^{\infty} p_i
\end{aligned}
$$

Now reindex the double sum to obtain

$$
\begin{aligned}
\eta &= \sum_{i=0}^{B-1} \sum_{j=i}^{B-1} (q_{\mathcal{P}}(j|i)p_i)\, j + B \sum_{i=0}^{B-1} q_{\mathcal{P}}(B \mid i)p_i + B \sum_{i=B}^{\infty} p_i \\
&= \sum_{i=0}^{B-1} p_i \sum_{j=i}^{B} q_{\mathcal{P}}(j|i)j + B \sum_{i=B}^{\infty} p_i .
\end{aligned}
$$

4. NUMBER OF PASSENGERS IN THE QUEUE

Although the queue is not a system (here we mean the queue itself as a system) for which the number in the system changes only by one, we can think of it being one as follows: when the passengers board as a batch, imagine that they leave the queue one by one (according to their order of arrival) but instantaneously. For such a system the stationary distribution of the number in the system as seen by the arrivals is the same as that as seen by the departures, and this is exactly the stationary distribution of the number in the system since the arrival process is Poisson [2].

Consider the moment at which the bus returns from a service and finds i passengers at the terminal. At most B of the passengers will board the bus, and the mth passenger who board will see (i–m) passengers behind him in the queue upon boarding. If $i \geq B$, the bus will depart with B passengers with probability 1. If $i \leq B-1$, the bus will depart with j passengers with probability $q_{\mathcal{P}}(j|i)$; the (j–i) passengers who arrive while the bus is waiting do not see any passenger behind upon boarding.

Let U be the number of passengers at the terminal seen by a typical passenger upon boarding. We will determine P[U=u] in two cases.

Case 1: $u = 0$

This is the case in which the passenger arrives at the terminal while the bus is waiting, or there are $1 \leq i \leq B$ passengers waiting when the bus arrives and the passenger is the last in the queue. Using Renewal Theory [3] we obtain

$$P[U=0] = \alpha \left[\sum_{i=0}^{B-1} p_i \sum_{j=i+1}^{B} q_{\mathcal{P}}(j|i)(j-i) + \sum_{i=1}^{B} p_i \right],$$

where α is the normalizing constant, which will be determined shortly.

Case 2: $u > 0$

This is the case in which there are $u+1 \leq i \leq u+B$ passengers waiting when the bus arrives, and the passenger is in the (i–u)th position in the queue. Therefore

$$P[U=u] = \alpha \left[\sum_{i=u+1}^{u+B} p_i \right].$$

The normalizing constant α is determined by

$$\begin{aligned}
\alpha^{-1} &= \left[\sum_{i=0}^{B-1} p_i \sum_{j=i+1}^{B} q_{\mathcal{P}}(j|i)(j-i) + \sum_{i=1}^{B} p_i \right] + \sum_{u=1}^{\infty} \left[\sum_{i=u+1}^{u+B} p_i \right] \\
&= \left[\sum_{i=0}^{B-1} p_i \sum_{j=i}^{B} q_{\mathcal{P}}(j|i)(j-i) + \sum_{i=1}^{B} p_i \right] + \sum_{u=1}^{\infty} \left[\sum_{i=u+1}^{u+B} p_i \right] \\
&= \sum_{i=0}^{B-1} p_i \sum_{j=i}^{B} q_{\mathcal{P}}(j|i)j - \sum_{i=0}^{B-1} p_i i \left[\sum_{j=i}^{B} q_{\mathcal{P}}(j|i) \right] + \sum_{i=1}^{B} p_i \\
&\quad + \sum_{u=1}^{\infty} \left[\sum_{i=u+1}^{u+B} p_i \right].
\end{aligned}$$

Using

$$\sum_{j=i}^{B} q_{\mathcal{P}}(j|i) = 1 ,$$

we get

$$\alpha^{-1} = \sum_{i=0}^{B-1} p_i \sum_{j=i}^{B} q_{\mathcal{P}}(j|i)j + \left[- \sum_{i=0}^{B-1} p_i i + \sum_{i=1}^{B} p_i + \sum_{u=1}^{\infty} \sum_{i=u+1}^{u+B} p_i \right] .$$

Now collect the coefficients of p_i in the parenthesis to obtain

$$\alpha = \left[\sum_{i=0}^{B-1} p_i \sum_{j=i}^{B} q_{\mathcal{P}}(j|i)j + B \sum_{i=B}^{\infty} p_i \right]^{-1} = \eta^{-1} .$$

Since the queue is FCFS, the stationary distribution of the waiting time in the queue is readily determined from the stationary distribution of the number of passengers in the queue [5].

5. SOJOURN TIMES

The sojourn time of a passenger is the time between his arrival at the queue and the time at which the bus he catches departs. Note that this in general is different from the waiting time in the queue because the passenger may wait on the bus. We will determine the stationary sojourn time distribution by considering a hypothetical process which is identical to the original process except the following. In the hypothetical process, the bus is always at the terminal. At the moment the bus in the original process departs for a service, the passengers on the bus in the hypothetical process leave the bus one by one instantaneously according to the order of their arrivals. Then the bus in the hypothetical process will not be available (that is, no passenger is allowed to board) until the bus in the original process returns from the service. Note the one–to–one correspondence between the passengers in the original and the hypothetical process. Now the queue and the bus in the hypothetical process together form a *system* for which the the number in the system changes only by one. Observe that the sojourn time of a passenger in the original process is exactly the time the passenger spend in *the system* in the hypothetical process. Using the same argument as that for the stationary distribution of the number of passengers in the queue, the stationary distribution of the number of passengers in the system as seen by the departures is exactly the stationary distribution. Then the stationary sojourn time distribution is readily determined from the stationary distribution of the number of passengers in the system, since the system is FCFS.

We will be referring to the original process unless otherwise mentioned. Now consider the bus returning from a service and find i passengers in the queue. If $i \geq B$, the bus will depart with B passengers with probability 1, and the mth passenger on the bus (assume the passengers retain their order after boarding the bus) will see (i–m) passengers behind upon leaving the system in the hypothetical process. If $i \le B-1$, the bus will depart with j passengers with probability $q_{\mathcal{P}}(j|i)$, and the mth passenger on the bus will see (j–m) passengers behind upon leaving the system in the hypothetical process.

Let V be the number of passengers in the system seen by a typical passenger upon leaving the system in the hypothetical process. We will determine P[V=v] in two cases.

Case 1: $0 \le v \le B-1$

This is the case in which the bus returns from a service and find $i \le B-1$ passengers in the queue, eventually departs with $v+1 \le j \le B$ passengers aboard, and the passenger is in the (j–v)th position on the bus, or there are $B \le i \le B+v$ passengers in the queue and the passenger is in the (i–v)th position. Now apply Renewal Theory to obtain

$$P[V=v] = \gamma \left[\sum_{i=0}^{B-1} p_i \sum_{j=v+1}^{B} q_{\mathcal{P}}(j|i) + \sum_{i=B}^{B+v} p_i \right].$$

Case 2: $v \ge B$

This is the case in which the bus returns from a service and find $v+1 \le i \le v+B$ passengers in the queue, and the passenger is in the (i–v)th position. Therefore

$$P[V=v] = \gamma \left[\sum_{i=v+1}^{v+B} p_i \right].$$

The normalizing constant γ is determined by

$$\begin{aligned}
\gamma^{-1} &= \sum_{v=0}^{B-1} \left[\sum_{i=0}^{B-1} p_i \sum_{j=v+1}^{B} q_{\mathcal{P}}(j|i) + \sum_{i=B}^{B+v} p_i \right] + \sum_{v=B}^{\infty} \left[\sum_{i=v+1}^{v+B} p_i \right] \\
&= \left[\sum_{v=0}^{B-1} \sum_{i=0}^{B-1} p_i \sum_{j=v+1}^{B} q_{\mathcal{P}}(j|i) \right] + \left[\sum_{v=0}^{B-1} \sum_{i=B}^{B+v} p_i + \right. \\
&\qquad \left. \sum_{v=B}^{\infty} \sum_{i=v+1}^{v+B} p_i \right] \\
&= \left[\sum_{i=0}^{B-1} p_i \sum_{v=0}^{B-1} \sum_{j=v+1}^{B} q_{\mathcal{P}}(j|i) \right] + \left[\sum_{v=0}^{B-1} \sum_{i=B}^{B+v} p_i + \right.
\end{aligned}$$

$$\left. \sum_{v=B}^{\infty} \sum_{i=v+1}^{v+B} p_i \right]$$

Upon collecting the coefficients of $q_{\mathcal{P}}(j|i)$ and p_i, we obtain

$$\gamma = \left[\sum_{i=0}^{B-1} p_i \sum_{j=i}^{B} q_{\mathcal{P}}(j|i)j + B \sum_{i=B}^{\infty} p_i \right]^{-1} = \eta^{-1} .$$

The significance of conditional departure distribution is now evident: it partitions $\mathcal{C}$ into equivalence classes of policies such that the stationary behavior of the queueing system under each policy within an equivalence class is identical.

6. REMARKS

It is worth pointing out that for a general $g(t)$, if the problem can be solved for a particular policy in $\mathcal{C}$ (possibly via some other scheme), it is a simple matter to determine $\{ p_i, i \geq 0 \}$. Since the p_i's do not depend on a particular policy, the problem can be solved for any other policy in $\mathcal{C}$ using the scheme we have presented. As a result, whether the problem is solvable or not is completely determined by the round trip time distribution $g(t)$.

ACKNOWLEDGMENT

Special thanks to B. Doshi, D. Jagerman, J. Kaufman, B. Sengupta and O.–C. Yue for their inspiring comments, which have made this work possible.

REFERENCES

[1] N. T. J. Bailey, "On queueing processes with bulk service," *J. R. Stat. Soc. Ser. B* 16, 80–87, 1954.

[2] L. Kleinrock, *Queueing Systems, Volume 1: Theory.* John Wiley & Sons, 1975.

[3] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume 1, 3rd Edition.* John Wiley & Sons, 1968.

[4] M. L. Chaudhry and J. G. C. Templeton, *A First Course in Bulk Queues.* John Wiley & Sons, 1983.

[5] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory, 2nd Edition.* John Wiley & Sons, 1985.

Optimal Scheduling of Parallel Processing Systems with Real-Time Constraints

François BACCELLI[1] , Zhen LIU[1], and Don TOWSLEY[2*]
1) INRIA Centre Sophia Antipolis 06565 Valbonne France
2) COINS, University of Massachusetts Amherst, MA 01003 U.S.A.

This paper focuses on the characterization of optimal scheduling policies in a general model of parallel processing systems, that was introduced in [1]. This initial model captured both the synchronization and the contention features that are characteristic of the parallel execution of structured jobs in multiprocessor systems. A simple mechanism has been added to it in order to represent real time constraints.

A job is composed of a set of tasks and a partial order specifying the precedence constraints between the tasks. We assume that all the jobs have the same task set and the same partial order, and that the task processing times are random variables with known probability distribution functions. The interarrival times of the jobs are also random variables with arbitrary distributions. The real time constraints are specified by reference times, also called soft real-time deadlines. We also assume that there is a predefined mapping from the set of tasks onto the set of machines, identical for all jobs, that allocates tasks to machines. The scheduling disciplines can be non-preemptive or preemptive-resume. In both cases, we assume that task processing times are independent and identically distributed. For preemptive disciplines, we assume in addition that they are exponentially distributed.

We are interested in such performance criteria as the number of jobs in the system, the throughput in term of jobs and the job lag times, where the lag time of a job stands for the difference between its reference time and its completion time. This extended abstract summarizes the optimality results on the subject obtained by the authors in [2].

We show first that policies that are First Come First Serve at the level of tasks (which implies First In First Out for jobs), minimize the number of jobs and maximize the system throughput in the strong stochastic ordering sense. We also show that such FCFS policies minimize the transient vector of response times in the Schur convex stochastic ordering sense, and minimize the stationary job response time in the convex stochastic ordering sense. Concerning real time applications, we prove that within the class of local order preserving policies that is defined in the paper, the Shortest Reference First and Longest Reference First policies respectively bound from below and from above the lag times vector of the n first customers, in the Schur convex stochastic ordering sense. This in turn characterizes the extremal values of the stationary distribution of the job lag times with respect to the convex ordering. In particular, it is shown that if the distribution function of the so called relative reference times is concave, then the stationary rate of jobs that meet their deadlines is maximal for the Last-in First-out (LIFO) policy and minimal for FCFS.

The basic tools for deriving these results rely on the theory of multivariate stochastic dominance and on simple interchange and coupling arguments. Besides their practical implications in real time parallel processing, the results that are derived extend several earlier results on classical queueing systems. The optimality results on the FCFS and LCFS policies can be viewed as a continuation of the convex ordering properties of the FIFO and LIFO service disciplines

---

[*] The work of this author was supported by NSF under grant ASC 88-8802764 and ONR under contract N00014-87-K-0796.

of Shantikumar and Sumita [3]. As to real time constraints, our results can also be seen as an extension of the optimality of the SRF policy that was established for tandem queues in [4].

References

[1] F. Baccelli, Z. Liu, *On the Execution of Parallel Programs on Multiprocessor Architectures, a Queueing Theory Approach.* INRIA Report No. 833, April 88, To Appear in the J. ACM.

[2] F. Baccelli, Z. Liu, and D. Towsley *Optimal Scheduling of Parallel Processing Systems with Real-Time Constraints.* COINS Technical Report, University of Massachusetts, 1989, submitted for publication.

[3] J. G. Shantikumar, U. Sumita, *Convex Ordering of Sojourn Times in Single-Server Queues: Extremal Properties of FIFO and LIFO Service Disciplines.* J. Appl. Prob., Vol. 24, pp.737-748, 1987.

[4] D. Towsley, F. Baccelli, *Comparison of Service Disciplines in a Tandem Queueing Network with Delay Dependent Customer Behavior*, COINS internal Report, July, 1989, submitted to Operations Research Letters.

A FAMILY OF JOB SCHEDULING STRATEGIES FOR AVERAGE RESPONSE TIME CONSTRAINTS

LEONIDAS GEORGIADIS, CHRISTOS NIKOLAOU, IOANNIS VINIOTIS

ABSTRACT

A priority queueing system with a single server and multiple customer classes is studied. The performance of each class is characterized by a prespecified response time objective. We develop a scheduling algorithm that meets those objectives, if they can be satisfied, or treats the classes "fairly", if they cannot. The algorithm possesses a number of desirable characteristics that makes it attractive for implementation: it is adaptive, requires minimal knowledge of job statistics (only arrival rates), and does not introduce excessive overhead, since its decisions are based on easily computable measures. Variants of the algorithm that possess additional features are also discussed.

I. INTRODUCTION

Priority queueing models can be successfully used to analyze and control complex computer systems in which different classes of users (customers) compete for a limited number of shared resources. The performance of a computer system can be influenced by three factors: (i) hardware characteristics (e.g. number of CPUs and I/O processors, memory speed, etc), (ii) demand characteristics (e.g. number and nature of different job types, arrival and service time patterns) and (iii) scheduling strategies (e.g. ordering of job execution, possibility of preemption, etc).

Most of the research effort on scheduling strategies has focused on the analysis of given priority policies such as head of the line priorities, round robin schemes etc [1]. Little attention has been given to design issues [2, 4, 10] and even less to the issue of finding policies that optimize certain performance criteria [5–8].

All those studies assume that certain statistical parameters (usually arrival and service time moments for the various job types) are known. This assumption is not always satisfied in practice. In many cases it requires extensive (and usually expensive) measurements, while often such statistics are not even available.

Lack of such information makes it also difficult to state reasonable performance objectives. Suppose for example that one of the objectives is to keep expected delays for a specific job type less than a threshold value g; it is obvious that this threshold should depend on the statistical parameters of the given class (e.g. it should be at least bigger than the average service time of this class.)

Since *a priori knowledge* of such statistics is not always available (or inexpensive), it is important to design *adaptive* priority policies that do not depend on such parameters. Adaptive algorithms are based on (continuous) monitoring of appropriate system variables. As information is collected, estimates of the unknown parameters become available and "adjustment" (convergence) is possible [3]. The amount of measurements required and the rate of convergence of the algorithm are two important characteristics for implementation purposes. The former is directly related to the overhead introduced by the algorithm, while the latter determines how fast the algorithm "learns" the unknown parameters or reacts to changes in the environment.

In this paper we propose adaptive algorithms that are appropriate for multiclass systems where average response time objectives are specified. The algorithms do not require knowledge of service time statistics, and can operate with low overhead. Algorithms that do not use the knowledge of arrival rates are also presented.

The remainder of the paper is organized as follows. In section II we present the queueing model that describes the workload statistics. In section III we formulate the optimization problem. In section IV we introduce an algorithm that solves it and discuss some of its characteristics. In section V we present (without proof) three variants of the given algorithm, that possess additional, desirable features.

II. THE QUEUEING MODEL

The system we consider consists of N classes of jobs (customers) and a single server. Customers that belong to class i arrive at the system according to a Poisson stream of rate of λ_i requests/sec. The arrival processes are statistically independent of each other. Customers not served upon arrival wait in an infinite capacity queue. The service time requirements for each class follow a distribution $B_i(x)$. Service times are also assumed independent of each other. When nonpreemptive priority policies are considered, we allow $B_i(x)$ to be an arbitrary probability distribution. However, when preemptive disciplines are introduced, we shall restrict our attention to exponential service time distributions.

III. THE OPTIMIZATION PROBLEM

The desired performance of each class i is characterized by a prespecified *response time objective* g_i. More specifically, for a given scheduling policy f, let $W_i(f)$ denote the (steady–state) average response time for requests of type i, when policy f is followed. Here the response time of a job is defined as the time the job spends in service plus the time it has to wait in the queue while other jobs are being served.

In general, a scheduling policy f can depend on the current and past queue lengths, on previous scheduling decisions, as well as on a randomization mechanism. We may also allow (or not) preemptions of the customer currently in service. We shall restrict ourselves to the set P of *work conserving* policies [12]. We call any member of the set P an *admissible* policy. It should be mentioned that even though P is a rich class of policies, it may not always contain the "optimal" one (for a given problem and a specified criterion of performance). This restriction, however, is inevitable since in many models (especially multiserver queueing models) the analysis becomes intractable otherwise.

Let us denote by

$$W(f) \triangleq (W_1(f), W_2(f), \cdots, W_N(f))$$

the vector of response times when policy f is in effect. A given response vector W is said to be achievable, if there exists an admissible policy f such that $W = W(f)$. The set of achievable response times obviously depends on the class of admissible policies. A thorough characterization of this set, for both preemptive and nonpreemptive scheduling policies is given in [10]. In [4] an extension for a multiserver queueing system is presented. We shall briefly outline the characterization of the achievable set for a single server system, since we shall use it later to describe the properties of our algorithms. In this presentation, we will consider the case where the service times have arbitrary distribution, and we will restrict attention to the class of work-conserving nonpreemptive policies. Similar results hold for work-conserving preemptive policies when the service times are exponentially distributed.

Let λ_i, μ_i denote the arrival and service rate of customers of class i, $i = 1, \cdots, N$. Define the utilization factor ρ_i of class i as $\rho_i \triangleq \lambda_i / \mu_i > 0$. In what follows, we make the following assumptions:

Assumption I: The system is stable, i.e., $\sum_{i=1}^{N} \rho_i < 1$.

Assumption II: The second moments of the service times of all classes are finite.

For the class of all (work-conserving) nonpreemptive policies, the space of achievable response times can be characterized by the following set of equations:

$$\sum_S \rho_i W_i = F_S, \quad S = \{1, 2, \cdots, N\}, \tag{1}$$

$$\sum_S \rho_i W_i \geq F_S, \quad S \subset \{1, 2, \cdots, N\}, \tag{2}$$

where $\{F_S\}$ are policy independent constants that depend on the statistical parameters of the queueing model. For more details and the exact forms of the constants, see [2]. For the purposes of our discussion, the value of these constants is not essential. Eq. (1) and (2) can be thought of as constraints that the performance of any scheduling algorithm must satisfy.

Our objective, is to find a policy π such that

$$W_i(\pi) \leq g_i, \quad i \in \{1, 2, \cdots, N\}, \tag{3}$$

for a given set of objectives $\{g_i\}$, where $g_i > 0$. Such a policy we shall call *optimal.* Clearly, if the given set of objectives, $\{g_i\}$, is too strict, there may be no policy that is optimal. On the other hand, if the objectives are achievable, more than one policies may be optimal.

Since a priori knowledge of the statistical parameters of the system is often inaccurate (or unavailable), it cannot be easily determined whether a set of objectives is satisfiable. The criterion in ineq. (3) does not suggest what actions should be taken when the objectives are not satisfiable by any policy. In practice, it is very important to have algorithms with contingency plans. Moreover, if the objectives are satisfiable by more than one policy, the criterion does not provide guidelines as to which policy to select among the set of optimal policies. To deal with the above mentioned problems, we consider the alternative criterion

$$\min_P \max_i \{\frac{W_i}{g_i}\} \tag{4}$$

The criterion in (4) is closely related to the criterion in eq. (3), for the following reasons:

(i) If there is a work conserving policy that satisfies criterion (3), the policy that satisfies criterion (4) also satisfies (3).

(ii) If the policy that satisfies criterion (4) does not satisfy criterion (3), no other work-conserving policy does.

(iii) The policy that satisfies criterion (4) is "fair" in the sense that the class with the worst performance is given the "most" attention. In fact, as we shall show, if equality of the ratios $\frac{W_i}{g_i}$ is achievable by any work-conserving policy, the policy that satisfies criterion (4) will achieve this equality.

The problem of constructing a policy that satisfies average response time objectives has been considered in [2, 8, 10, 11]. The algorithms proposed in the above mentioned references are not adaptive and rely heavily on knowledge of the statistical parameters of the system. Moreover, they require solution of a large number of equations. Even for moderate values of N, the numerical solution of these equations becomes very cumbersome, if at all possible. Their implementation is, therefore, questionable.

Our purpose, as we describe next, is to design *adaptive*, *low overhead* algorithms that require minimal a priori knowledge of statistical parameters and satisfy criterion (4).

The basic idea is to monitor the performance of the various classes and adjust the priority levels according to "worst current" performance. This is in contrast to the basic philosophy behind most of the present algorithms (e.g., [7, 8, 10]). Those algorithms assign priority levels randomly, according to a probability distribution. Nonadaptiveness and excessive computations required to compute the probability distribution, are fundamental drawbacks of the above mentioned algorithms.

Our algorithms are dynamic, akin to the time dependent priority rules described in [1]. The difference is that the priority of a class can vary more dynamically, adapting to the current status of the performance of the class.

The algorithms we present (here and in section V) make *explicit* use of the history of the system. To the best of our knowledge, the majority of the scheduling algorithms presented in the literature are either purely Markovian, or they possess regenerative points (usually each decision instant is such a point). As we shall see, the lack of such regeneration instants for our algorithms complicates considerably the proof of ergodicity.

More specifically, let $\eta_i(t)$ be the queue size (including the customer currently being served, if any) of class i at time t. Also at time T, let

$$P_i(T) \stackrel{\Delta}{=} \frac{1}{\lambda_i g_i} \int_0^T \eta_i(t) dt$$

be the "priority index" of class i, $i = 1, \cdots, N$. The algorithms we describe in this paper, base the priority assignments on this index.

Algorithm A1 is the "basic" algorithm. In section V a number of variants are proposed.

Algorithm A1: Let t_k be the beginning of the kth busy period. At time t_k, sort $P_i(t_k)$ in decreasing order. For the duration of busy period k, give (nonpreemptive) priority to class j over class i, iff

$$P_j(t_k) > P_i(t_k)$$

Resolve ties in an arbitrary manner, e.g., by throwing a random coin.

Observe that the priority indices can be expressed as

$$P_i(t_{k+1}) = P_i(t_k) + \frac{1}{\lambda_i g_i} \int_{t_k}^{t_{k+1}} \eta_i(t) dt$$

Therefore it is easy to compute them recursively, by observing the history of the system. Most computer installations take measurements of the evolution of the system at arrival and service completion instants. Therefore, the computation of the indices most likely will not introduce the need for extra measurements. Moreover, the integral is easily computable, since the queue sizes are step functions. Thus, in most cases, the overhead introduced by the algorithm comes from the need to sort the vector of priority indices.*

---

* Compare this overhead to the computational effort required to solve a linear program with $N!$ inequalities, as discussed in [8].

In the next section we show that when the system is ergodic, algorithm A1 (and, in fact, any of the variants in section VI,) satisfies criterion (4).

IV. OPTIMALITY OF THE ALGORITHMS AND ERGODICITY

Let π be one of the above mentioned policies (i.e., algorithm A1 or one of its variants). Let $w_i(k), k = 1, 2, \cdots$ be the response time of the kth customer from class i, when algorithm π is employed. Assume that π produces an ergodic system. In other words, for $i = 1, \cdots, N$, the limits

$$\lim_{k\to\infty} \frac{\sum_{l=1}^{k} w_i(l)}{k}$$

exist a.e. (and are equal to W_i, as defined in section III). In this section, we show that the response times W_i satisfy criterion (4). We need a preliminary lemma first. Its proof may be found in [13].

Lemma 1: Let $\{F_S\}$ be the constants appearing in ineq (1), (2). The problem

$$\min_{x_i} \max\{x_1, \cdots, x_N\} \tag{5}$$

$$\sum_S c_i x_i \geq F_S \qquad S \subseteq \{1, 2, \cdots, N\}$$

$$x_i \geq 0, \quad c_i > 0,$$

has a solution. Its optimal value is given by $c = \max_S \frac{F_S}{\sum_S c_i}$.

We are now ready to state the main result of this paper. As we mentioned earlier, the proof makes use of the fact that the optimal policy is ergodic.

Proposition 1: Suppose that algorithm π is ergodic. Then it is optimal for the criterion in eq. (4).

The proof of this proposition may be found in [13]. While it is intuitive that the system is ergodic, the proof presents some analytical difficulties. We present here the proof for two classes of customers. Some rather technical details are given in [13].

Let t_k denote the start of the kth busy period. Define further

$$X(k) \overset{\Delta}{=} P_1(t_k) - P_2(t_k), \tag{6.1}$$

$$\delta_i(k) \overset{\Delta}{=} \frac{1}{\lambda_i g_i} \int_{t_k}^{t_{k+1}} \eta_i(t)dt, \quad i = 1, 2, \tag{6.2}$$

$$\Delta(k) \overset{\Delta}{=} \delta_1(k) - \delta_2(k). \tag{6.3}$$

Let $m_i(k)$, $i = 1, 2$, be the number of customers from class i, served during the kth busy period. Let

$$M_i(k) \overset{\Delta}{=} \sum_{l=1}^{k} m_i(l), \quad i = 1, 2,$$

be the total number of class i customers served in the first k busy periods. Since the system is stable and the policies are work conserving, the sequence $m_i(k)$, $k = 1, 2, \cdots$, is a sequence of i.i.d. random variables, with finite mean. This observation is instrumental in the ergodicity proof presented here. As an immediate consequence, we have that $M_i(k)/k$ converges a.e. to a positive, finite constant d_i (which is equal to the mean of $m_i(k)$).

From the definition of the priority indices and the response time for a typical customer (see section III), we have that

$$P_i(t_k) = \frac{1}{\lambda_i g_i} \sum_{l=1}^{M_i(k)} w_i(l), \quad i = 1, 2. \tag{7}$$

From eq (7) we will see that in order to study ergodicity (i.e., the behavior of the limits $\lim_{n\to\infty} \sum_{l=1}^{n} w_i(l)/n$), it suffices to determine the behavior of $P_i(t_k)/k$.

For a fixed value of k, and for $i = 1, 2$ define

$$E_i(k) \stackrel{\Delta}{=} \{l : 1 \le l \le k, \text{ class i has highest priority during busy cycle l}\}.$$

Obviously $\{1, 2, \cdots, k\} = E_1(k) \cup E_2(k)$. Denote the cardinality of the set E_1 (resp. E_2) as $N_1(k)$ (resp. $N_2(k)$). In other words, $N_1(k)$ represents the number of busy periods (among the first k) over which class 1 has highest priority.

Let us further define* the following "drifts":

$$A^+ \stackrel{\Delta}{=} E(\Delta(k+1) | X(k) > 0),$$

$$A^- \stackrel{\Delta}{=} E(\Delta(k+1) | X(k) < 0).$$

The limiting behavior of $X(k)$ and $N_i(k)$ is described in theorem 1, in terms of the drifts we defined above.

Theorem 1: There are four mutually exclusive possibilities, each taking place a.e.:

(i) $A^+ > 0, A^- > 0$: In this case, $\lim_{k\to\infty} X(k)/k = A^+$, and

$$\lim_{k\to\infty} N_1(k) = \infty, \quad \lim_{k\to\infty} N_2(k) < \infty.$$

(ii) $A^+ \le 0, A^- > 0$: In this case, $\lim_{k\to\infty} X(k)/k = 0$, and

$$\lim_{k\to\infty} N_1(k) = \infty, \quad \lim_{k\to\infty} N_2(k) = \infty,$$

$$\lim_{k\to\infty} \frac{N_1(k)}{k} = \frac{A^-}{A^- - A^+} \quad \lim_{k\to\infty} \frac{N_2(k)}{k} = \frac{-A^+}{A^- - A^+}.$$

(iii) $A^+ < 0, A^- = 0$: Same as case (ii).

(iv) $A^+ < 0, A^- < 0$: In this case, $\lim_{k\to\infty} X(k)/k = A^-$, and

$$\lim_{k\to\infty} N_1(k) < \infty, \quad \lim_{k\to\infty} N_2(k) = \infty.$$

The proof of the theorem is given in [13]. A geometric interpretation, that clarifies the methodology of the proof and highlights the performance of the system will be given at the end of this section.

---

* One should expect A^+ and A^- to depend on $k, X(k)$. It is easy to see, however, that they do not.

Lemma 2 describes the limiting behavior of the priority indices. Its proof is based on theorem 1.

Lemma 2: $P_i(t_k)/k$ converges a.e. to a constant r_i.

It follows then immediately from (7) that

$$r_i = \frac{1}{\lambda_i g_i} \lim_{k\to\infty} \frac{\sum_{l=1}^{M_i(k)} w_i(l)}{k} = \lim_{k\to\infty} \frac{\sum_{l=1}^{M_i(k)} w_i(l)}{M_i(k)} \cdot \frac{M_i(k)}{\lambda_i g_i k}, \ a.e., \quad i = 1,2, \tag{8}$$

and thus,

$$\lim_{k\to\infty} \frac{\sum_{l=1}^{M_i(k)} w_i(l)}{M_i(k)} = \frac{r_i \lambda_i g_i}{d_i}, \ a.e. \tag{9}$$

Eq. (9) is instrumental in the proof of proposition 2.

Proposition 2: Algorithm A.1 produces an ergodic system, i.e.,

$$\lim_{n\to\infty} \frac{\sum_{l=1}^{n} w_i(l)}{n} = W_i, \ a.e., \quad i = 1,2,$$

where W_i is the average (steady state) response time for class i.

Theorem 1 has an interesting geometric interpretation that highlights the behavior of the system under algorithm A1 (see figure 1). On the plane (W_1, W_2), the set of achievable average response times for work conserving policies is a line segment (segment EG in figure 1). Let $\bar{W}_i$, $i = 1,2$, be the average response time when class 1 has always highest (nonpreemptive) priority. Then we have that $E = (\bar{W}_1, \bar{W}_2)$. Let $\bar{B}$ be the average length of a busy period. Then $\lambda_i \bar{B}_i$, $i = 1,2$, is the average number of customers from class i that are served during a busy period. Also, $E\{\delta_i(k)|X_k > 0\}$, $i = 1,2$, is the average cumulative response time of customers from class i when class 1 has higher priority. It follows from the regenerative theorem [9, prop. 3.7.2], that

$$\frac{E\{\delta_i(k)|X_k > 0\}}{\lambda_i \bar{B}} = \bar{W}_i, \quad i = 1,2. \tag{10}$$

From equation (10) and the definition of A^+, A^- we conclude that

$$\frac{A^+}{\bar{B}} = \frac{\bar{W}_1}{g_1} - \frac{\bar{W}_2}{g_2}. \tag{11}$$

On the plane (W_1, W_2), consider the line $l: \ W_1/g_1 = W_2/g_2$. We distinguish the following cases:

(1). $A^+ > 0$. From equation (11) we see that in this case $\bar{W}_1/g_1 > \bar{W}_2/g_2$, and therefore line l does not intersect the segment EG. Point E lies "below" line l (line #1 in figure 2). Equality of ratios cannot be achieved in this case. From part (i) of theorem 1, we see that after a finite time, highest priority is always given to class 1. Since class 1 has the larger ratio W/g, we immediately see that the algorithm is "fair", in the sense that it tries to minimize the larger of the two ratios.

(2). $A^+ = 0$. In this case, we have that $\bar{W}_1/g_1 = \bar{W}_2/g_2$, and line l intersects segment EG at the end point E. Equality of ratios is achieved in this case. From theorem 1, part (ii), we see that class 2 is given higher priority infinitely many times. However, the proportion of time that class 2 has higher priority, is zero.

(3). $A^+ < 0$. In this case, $\bar{W}_1/g_1 = \bar{W}_2/g_2$, and line l intersects segment EG at the "interior" point F. Equality of ratios is achieved. From part (ii) of theorem 1, priorities alternate between the two classes infinitely many times. Moreover, the proportion of time during which class i, $i = 1, 2$, has higher priority, is strictly positive.

V. VARIANTS OF THE BASIC ALGORITHM

In this section we present some variants of the basic algorithm. We claim, without any proof, however, that they produce ergodic systems. They represent a "trade off" between overhead and rate of convergence.

Algorithm A2 takes decisions "more frequently". The amount of overhead is adjusted by the "period" C.

Algorithm A2: Let C be a positive constant and consider the time instants $t_m = mC$, $m = 1, 2, \cdots$. Let t_k be the beginning of a busy period, or one of the instants t_m whichever comes first. At time t_k sort $P_i(t_k)$ in decreasing order. For the duration of the interval until the end of the busy period, or the next instant t_m, whichever comes first, give nonpreemptive priority to class j over class i, iff

$$P_j(t_k) > P_i(t_k).$$

Resolve ties in an arbitrary manner, e.g. by throwing a random coin.

Algorithm A3 introduces the largest amount of overhead. It could be used, for example, when a faster rate of convergence is desired, or when traffic is not too heavy.

Algorithm A3: Let t_k be the beginning of a busy period or the instant of service completion whichever comes first. At time t_k, serve a customer from the class with the highest index $P_i(t_k)$. Resolve ties in an arbitrary manner, e.g. by throwing a random coin.

As expressed, algorithms A1–A3 require only the knowledge of class arrival rates. We can easily modify them, however, so that even this knowledge is not required. Let

$$\lambda_i(t) \triangleq \frac{\text{number of class i arrivals in interval } [0, t)}{t}$$

and

$$P_i^{'}(t_k) \triangleq \frac{1}{\lambda_i(t_k) g_i} \int_0^{t_k} \eta_i(t) dt.$$

Any of the previous algorithms can be applied by using $P_i^{'}(T)$ in place of $P_i(T)$.

Once existence of steady state response times, under these algorithms is established, their optimality (for criterion (4)) is immediate, since the proof of proposition (1) remains the same.

CONCLUSIONS

We have considered a multiclass system where predefined performance objectives are to be satisfied. We have provided an algorithm that achieves the specified objectives if they can be satisfied and treats the various workload classes "fairly", if they cannot. The main advantages of the proposed algorithm, over existing ones, are: adaptability, lack of a priori computations, insensitivity to statistical measurements and fairness.

A number of nontrivial extensions and complementations of this work seem worth pursuing. A rigorous proof of ergodicity for algorithms that *explicitly* depend on past system behavior is definitely an issue of theoretical importance. In addition, the rate of convergence of the algorithms is an issue of both theoretical and practical importance. In some applications the rate of convergence may be critical. The model we studied assumed availability of one server only. In practice, systems with more than one CPU (and independent I/O processors, memory managers etc.) are abundant. Conservation laws, that are instrumental here, do not exist for such systems. Therefore, alternative characterizations of "good" policies for such systems, are of paramount (theoretical and practical) importance.

REFERENCES

[1] L. Kleinrock, "Queueing Systems, vol. 2: Computer Applications," John Wiley, 1976.

[2] E. Gelenbe–I. Mitrani "Analysis and Synthesis of Computer Systems," Academic Press, New York, 1980.

[3] A. Schwartz–A. Makowski, "An Optimal Adaptive Scheme for Two Competing Queues with Constraints," Proc. 7th Int INRIA Conf. Anal. Optim. Syst., 1986. pp. 515–532.

[4] A. Federgruen–H. Groenevelt, "Characterization and Optimization of Achievable Performance in General Queueing Systems," Oper. Res., vol. 36, 1988, pp. 733–741.

[5] J. M. Harrison, "Dynamic Scheduling of a Multiclass Queue: Discount Optimality," Oper. Res., vol. 23, pp. 270–282, 1975.

[6] G. P. Klimov, "Time Sharing Service Systems I," Theory Prob. Applications, vol. 19, pp. 532–551, 1974.

[7] K. W. Ross–B. Chen, "Optimal Scheduling of Interactive and Noninteractive Traffic in Telecommunication Systems," IEEE Transactions on Automatic Control, vol. AC–33, no 2, pp. 261–267, 1988.

[8] K. W. Ross–D. D. Yao, "Optimal Dynamic Scheduling in Jackson Networks," IEEE Transactions on Automatic Control, vol. AC–34, no 1, pp. 47–53, 1989.

[9] S.M. Ross, "Stochastic Processes," Academic Press, 1983.

[10] E. G. Coffman–I. Mitrani, "A Characterization of Waiting Time Performance Realizable by Single Server Queues," Operations Research, vol. 28, pp. 810–821, 1980.

[11] I. Mitrani and J. H. Hine, "Complete Parametrized Families of Job Scheduling Strategies," Acta Informatica, Vol 8, pp. 61-73, 1977.

[12] D. P. Heyman, M. J. Sobel, "Stochastic Models in Operations Research," vol I, McGrow-Hill, New York 1982.

[13] L. Georgiadis, C. Nikolaou, I. Viniotis, "Adaptive Scheduling Algorithms that Satisfy Average Response Time Objectives," IBM Technical Report, Yorktown Heights, 1989.

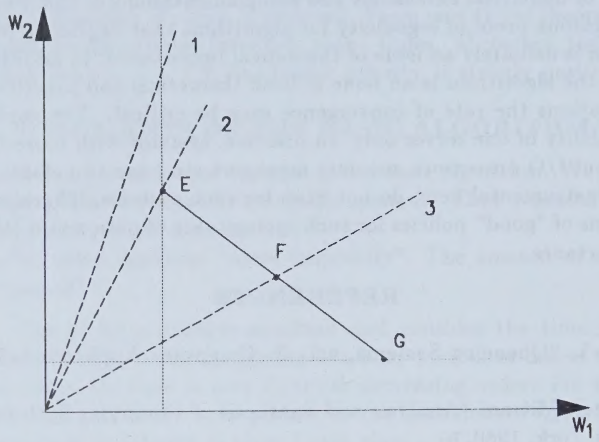

W2
1
2
E
3
F
G
W1

Figure 1. Geometric Interpretation of system behavior.

OPEN-LOOP FEEDBACK POLICIES FOR MARKOVIAN QUEUEING SYSTEMS

RODOLFO A. MILITO
AT&T Bell Laboratories, Holmdel, N.J. 07733
rodolfo%houxa@research.att.com

ABSTRACT

This paper recasts a recent approach to the control of Markovian systems proposed by Krishnan and Ott[1] in the framework of open-loop feedback control policies and the value iteration algorithm. A simple recursion for the differential costs of a queue in isolation is obtained as a limit of the recursion for the discounted cost-to-go. This allows the treatment of general costs, state-dependent servers, and limited buffers. Several possible extensions of the approach are also discussed.

1. INTRODUCTION

There is a recognized need for efficient, implementable control strategies in computer communication networks, including routing, scheduling and admission control policies.

Markovian decision processes have proven appropriate to model these problems. This methodology has been extensively used to characterize structural properties[2] [3] [4] of the optimal strategies, thus providing useful insight about the system behavior. Unfortunately, the Markovian decision process paradigm falls short of providing implementable rules except for the simplest problems. The Markovian decision approach is not conducive to closed form solutions. Even worse, the dimensionality of the state space rules out numerical solutions except for small sized systems. Hence the quest for approximations that, at the expense of some degree of optimality, yield implementable rules. Motivated by the routing problem in circuit-switched networks, Krishnan and Ott[1] proposed the following approach:

i) Obtain the optimal stationary static routing policy (Bernouilli splitting). ii) Derive the differential cost-to-go (relative Howard costs) for each trunk group in isolation. iii)Use the above to determine an explicit rule as the outcome of the first step of the policy iteration algorithm.

In a similar vein, Milito and Levy[5] proved some structural properties of the optimal policy for a blocking system, and used the insight thus provided to derive a heuristic rule with near-optimal performance. Milito and Levy[5] used a discounted model, and their heuristic rule required the derivation of the cost-to-go of a queue in isolation. This was done by solving a recurrence equation, rather than resorting to queueing theoretic arguments used by Krishnan and Ott[1]. Krishnan and Ott's approach has been extended to queueing systems with delay and linear costs[6] and blocking[7]. The queueing theoretic machinery, successfully used by Krishnan to derive the differential cost-to-go when the cost is a linear function of the number in queue, or when blocking is the sole performance measure of interest, does not lend itself to an easy generalization to more general cost structures.

In a recent internal AT&T Symposium, Milito and Cudjoe[8] advanced an alternative approach, based on the limit of a recursion that represents the dynamics of the discounted cost-to-go of a queue in isolation, when the discount factor tends to one. That talk focused on the problem of routing impatient customers belonging to different classes to a set of possible destinations, modeled as parallel queues. The cost, a highly non-linear function of the queue length, intended to capture the customer abandonment behavior. The purpose of this paper is to present this methodology in a more general framework to a wider audience.

The rest of the paper is organized as follows. Section 2 motivates the interest in non-linear cost structures. Section 3 reviews pertinent results from the theory of Markovian Decision Processes. Open-loop-optimal feedback control policies are discussed in section 4 in the context of Markovian queueing systems, with particular reference to two problems: routing to queues in parallel, and scheduling of a server among parallel queues. The paper ends with section 5, where some open problems are outlined.

2. MOTIVATION FOR NON-LINEAR COST STRUCTURES

Little's result[9] justifies the emphasis put on cost structures that are a linear function of the number in the queue, since *waiting time* is a performance measure of interest. If the control law does not affect the arrivals, the minimization of the number in the system corresponds to the minimization of the mean sojourn time. In certain situations, however, a linear cost structure cannot capture the full complexity of the system. This is the case with impatient customers. A thorough characterization of customer abandonment behavior is not an easy task, but a crude heuristic approximation can be made along the following reasoning[8]. The probability of abandonment is a non-linear function of the waiting time, that remains flat up to a point in which it increases rapidly, to flatten again when the probability approaches 1. Just because of the abandonments, there is no one to one correspondence between number in the queue and delay. However, if the total abandonment rate is not exceedingly high, such a correspondence can be assumed.

Finite buffers provide another instance where non-linear costs are useful. The problem of analyzing a queueing network with finite buffers being difficult, that of designing good dynamic routing rules in such a network becomes overwhelming. Non-linear costs allow us to model the network as one with infinite buffers that incurs in arbitrarily high penalty when certain queuing thresholds are exceeded.

3. MARKOVIAN DECISION MODELS: SOME PERTINENT RESULTS

Consider the following discrete-time Markov decision model. Let S be a finite state space, and, for each state $x \in S$, let $A(x)$ be a finite set of admissible decisions. When the state of the system is $x \in S$, and decision $a_x \in A(x)$ is taken, the system incurs in an immediate cost $c_x(a_x)$ and jumps, in the next epoch, to state y with probability $p_{xy}(a_x)$, where $\sum_{y \in S} p_{xy}(a_x)=1$. A stationary policy π is a mapping $S \to A = \bigcup A(x)$. Let $g(\pi)$ denote the long run average cost incurred by the system under policy π, and $a_x^{\pi}=\pi(x)$ the decision taken under policy π when the system is in state x. The following results will be useful in the sequel.

Theorem 3.1:

Let g, $\{v_x, x \in S\}$, be given numbers. Suppose that for a policy π the following set of inequalities holds

$$c_x(a_x) - g + \sum_{y \in S} p_{xy}(a_x) v_y \leq v_x \quad \textit{for each } x \in S. \tag{3.1}$$

Then,

$$g(\pi) \leq g. \tag{3.2}$$

The claim also holds when the inequalities in (3.1) and (3.2) are reversed.

Proof: See, e.g., Tijms[10].

□

The following theorem plays a central role in the value iteration algorithm.

Theorem 3.2:

Suppose that for any stationary policy π there exists a state s (possibly dependent on on π) that can be reached from any other state under policy π. (Unichain assumption). Let $\xi = (\xi_x, x \in S)$ be an arbitrary vector, $T\xi$ a new vector computed according to

$$T\xi_x = \min_{a \in A(x)} \{c_x(a) + \sum_{y \in S} p_{xy}(a) \xi_y\}, \quad \textit{for each } x \in S, \tag{3.3}$$

and π the policy that minimizes the right hand side of (3.3). Then,

$$m \triangleq \min_{x \in S} \{T\xi_x - \xi_x\} \leq g^* \leq g(\pi) \leq \min_{x \in S} \{T\xi_x - \xi_x\} \triangleq M,$$

where $g^* \triangleq \min_{\pi} g(\pi)$. *Proof:* See Tijms[11].

□

To introduce notation that will be useful in the sequel, we recall here the

Value iteration algorithm

Step 0. Choose $\{ v_x^{(0)}, x \in S\}$ with $0 \leq v_x^{(0)} \leq \min_a c_x(a)$ for all $x \in S$. Let $n := 1$.

Step 1. Compute the value function $v_x^{(n)}$, $x \in S$, from

$$\begin{aligned} v_x^{(n)} &= \min_{a \in A(x)} \{c_x(a) + \sum_{y \in S} p_{xy}(a) v_y^{(n-1)}\} \\ &\triangleq c_x(a_x^{(n)}) + \sum_{y \in S} p_{xy}(a_x^{(n)}) v_y^{(n-1)}\} \end{aligned} \tag{3.4}$$

and denote as $\pi^{(n)}$ a stationary policy that minimizes the right hand side of (3.3), i.e., $a_x^{(n)} = \pi^{(n)}(x)$.

Step 3: Compute the bounds

$$m_n \triangleq \min_{x \in S} \{v_x^{(n)} - v_x^{(n-1)}\} \quad \textit{and} \quad M_n \triangleq \max_{x \in S} \{v_x^{(n)} - v_x^{(n-1)}\}$$

Stop with policy $\pi^{(n)}$ if $M_n - m_n \leq \varepsilon\, m_n$. Otherwise, go to step 3.

Step 3: n:= n+1 and go to step 1.

The stopping criterion is based on the following corollary of theorem 3.2:

Corollary of theorem 3.2:[10]

$\{m_n\}$ is not decreasing, $\{M_n\}$ is not increasing, and

$$\frac{M_n - m_n}{m_n} \leq \varepsilon \textit{ implies } \frac{g(\pi^{(n)}) - g^*}{g^*} \leq \varepsilon.$$

□

$v_x^{(n)}$ can be interpreted as the minimal total expected cost for a time horizon of n periods when the current state is x and a terminal cost $v_y^{(0)}$ is incurred if the system ends up in state y. Note that in this interpretation the choice of the vector of terminal costs $\{v_x^{(0)}, x \in S\}$ is arbitrary. The next theorem examines the situation when $\{v_x^{(0)}, x \in S\}$ is a vector of relative values associated with a policy $\pi^{(0)}$.

Theorem 3.3:

Suppose that for a given admissible policy $\pi^{(0)}$ the following equalities hold

$$v_x^{(0)} = c_x(a_x^{(0)}) - g^{(0)} + \sum_{y \in S} p_{xy}(a_x^{(0)}) v_y^{(0)}, \quad \textit{for each } x \in S, \tag{3.5}$$

where $a_x^{(0)} = \pi^{(0)}(x)$. Then, the value iteration algorithm yields

$$g(\pi^{(n)}) \leq g^{(0)} - \delta^{(n)}, \tag{3.6}$$

where $\delta^{(0)} = 0$, and $\delta^{(n)} \geq \delta^{(n-1)}$ for all n.

Proof:

□

Define

$$\delta^{(n)} = \min_{x \in S} \sup_{\delta} \{\delta : v_x^{(n)} - v_x^{(n-1)} \leq g^{(0)} - \delta\}. \tag{3.7}$$

Claim: $\delta^{(n)}$ is monotonically non-decreasing in n.

Proof of the claim: It follows from (3.4) and (3.5) that

$$v_x^{(1)} \leq v_x^{(0)} + g^{(0)}.$$

Let us assume that the claim holds for $n > 1$. Then

$$v_x^{(n+1)} \leq c_x(a_x^{(n)}) + \sum_{y \in S} p_{xy}(a_x^{(n)}) v_y^{(n)} \leq c_x(a_x^{(n)}) + \sum_{y \in S} p_{xy}(a_x^{(n)}) [v_y^{(n-1)} + g^{(0)} - \delta^{(n)}],$$
$$= v_x^{(n)} + g^{(0)} - \delta^{(n)} \quad \textit{for each } x \in S.$$

The validity of the claim follows from the definition of $\delta^{(n)}$.

Proof of the theorem:

$$v_x^{(n+1)} = c_x(a_x^{(n+1)}) + \sum_{y\in S} p_{xy}(a_x^{(n+1)})v_y^{(n)} \le v_x^{(n)} + g^{(0)} - \delta^{(n+1)}.$$

From theorem 3.1 we conclude that

$$g(\pi^{(n+1)}) \le g^{(0)} - \delta^{(n+1)}.$$

□

To link these results to the next section that deals with open-loop feedback policies, we need an interpretation of the policy generated by the the value iteration algorithm initialized according to (3.5). Towards this goal, we recall a few known facts. It follows from theorem (3.1) that the value $g^{(0)}$ defined by (3.5) is the long run average incurred by the system under policy $\pi^{(0)}$. Also, $v_x^{(0)} - v_y^{(0)}$ gives the difference in total expected costs over and infinite horizon for starting in state x rather than in state y, given that policy $\pi^{(0)}$ is followed. This is an immediate consequence of the fact that, for m large enough,

$$V_x^{(0)}(m) \approx mg^{(0)} + v_x^{(0)}, \tag{3.8}$$

where $V_x^{(0)}(m)$ stands for the expected cost over an m-periods horizon when the system starts at state x and policy $\pi^{(0)}$ is followed.

Consider now $C^{\pi^0}(n,m)$, the class of non-stationary policies determined by the following conditions: (a) During the first n stages, decisions are only restricted by the admissibility set A. (b) Thereafter, i.e., from period $n+1$ to period m, policy $\pi^{(0)}$ is followed.

Let $\gamma^{(n)}(\pi^0)$ denote the following non-stationary strategy generated by n iterations of the value iteration algorithm initialized according to (3.5): apply policy $\pi^{(n-i)}$ at stages i, $i = 0,1, \cdots n$, and policy $\pi^{(0)}$ thereafter. Now denote as $\gamma(m)$ the policy that minimizes the total expected cost within the class $C^{\pi^0}(n,m)$, i.e.,

$$\gamma(m) = arg \min_{C^{\pi}(n,m)} V_x^{\gamma}(n+m), \tag{3.9}$$

where $V_x^{\gamma}(n+m)$ gives the expected cost over $n+m$ periods when policy γ is followed. Then, by virtue of (3.4) and (3.8), we conclude that

$$\lim_{m\to\infty} \gamma(m) = \gamma^{(n)}(\pi^0). \tag{3.10}$$

Therefore, the sequence of decisions prescribed by n iterations of the value iteration algorithm initialized according to (3.5) minimize the expected cost of the system that, after the n^{th} stage, evolves according to the policy $\pi^{(0)}$. The stationary policy $\pi^{(n)}$ is an *open-loop feedback policy of vision n*, according to the definition we introduce in the following section.

4. OPEN-LOOP OPTIMAL FEEDBACK CONTROL POLICIES

4.1 General definitions of open-loop feedback controllers (OLFC)

An open-loop optimal feedback controller operates as follows:[12] [13] At each decision epoch k, $k = 0,1, \cdots$, uses the available information to compute the entire future control sequence, as if no further measurements were to be received in the future. It then applies the control action prescribed by the first element in the control sequence, and repeats the procedure at the next decision epoch. One can also define an *open-loop controller of vision n* as the one that, in its computation of the control sequence, acknowledges that $n-1$ more state measurements will become available.

4.2 OLFC in the Markovian decision process framework

The prescription of an OLFC of *vision n* in this context is

(a) Choose $\pi^{(0)}$ as an optimal static policy, and find $\{v_x^{(0)}, x \in S\}$ that satisfy (3.5).

(b) Compute $\pi^{(n)}$ as the n^{th} step of the value iteration algorithm.

(c) Use $\pi^{(n)}$ as the the OLFC of *vision n*.

Theorem 3.3 guarantees that the policy thus derived is not outperformed by the optimal static one. In the case $n=1$, i.e., the standard OLFC, the first step of the value iteration algorithm coincides with the policy iteration algorithm. This was the standpoint of Krishnan and Ott[1]. Policy iteration requires the solution of a system of linear equations at each iteration, which might be unfeasible for large systems. Although the value iteration algorithm only guarantees a monotonic improvement of the bounds with the number of iterations it is computationally more attractive. It will be useful in the sequel to note that that (3.4) can be rewritten in terms of the differential costs as follows:

$$v_x^{(n)} - v_y^{(n)} = v_x^{(n-1)} - v_y^{(n-1)} + \min_{a \in S} \{ c_x(a) + \sum_{z \in S} p_{xz}(a)(v_z^{(n-1)} - v_x^{(n-1)})\} \tag{4.1}$$
$$- \min_{a \in S} \{ c_y(a) + \sum_{z \in S} p_{yz}(a)(v_z^{(n-1)} - v_y^{(n-1)})\}.$$

4.3 OLFC in the context of Markovian queueing systems

We consider a Markovian queueing system whose performance behavior is captured by the sum of (possible) non-linear functions of the number of customers in each queue. The goal is to derive a control policy that, in the long run, nearly optimizes the system performance. Typical control actions include routing of customers, allocation of servers, and admission or rejection of customers. The approach is summarized in the following steps:

1. Formulation of the Markovian decision process model, using the *uniformization* technique[14] to obtain a discrete time representation.

2. Formulation of the optimal Open-loop controller (OLC) policy, that has the virtue of decoupling the dynamics of the different queues.

3. Derivation of a recursion for the differential cost-to-go for a Markovian queue in isolation, i.e., the long term difference in the cost-to-go for starting the queue with $i+1$ rather than i customers. Details of the derivation are given in the appendix.

4. Derivation of the Open Loop Feedback Controller (OLFC) of *vision n* [13] as n^{th} step of the policy iteration algorithm based on the optimal OLC. As noted before (recursion 4.1), this only involves the differential costs.

The application of the approach is straightforward for the following problems:

(a) Customers that arrive in a Poisson stream of intensity λ are to be routed to one of N queues, $Q_1, Q_2, \cdots Q_N$. Queue Q_i, $i=1,2,..N$ has a state dependent exponential server that operates at rate $\mu_i(x_i)$, where x_i is the number of customers in the queue. The state information $x = (x_1, \cdots, x_N)$ is available, and the criterion is to minimize the long run average cost, where the immediate cost is given by $\sum c_i(x_i)$, for arbitrary functions $c_i(\cdot)$. Both infinite and limited buffer sizes can be handled.

(b) The work of a single exponential server is to be scheduled among N queues. Jobs that require $1/\mu_i$ seconds of service time arrive to queue Q_i in a Poisson stream of intensity λ_i. The problem is to schedule the server so that the expected long run average cost is minimized, the immediate cost being $\sum c_i(x_i)$, for arbitrary functions $c_i(\cdot)$, where x_i is the number in Q_i. Preemption is available, and buffers can be either limited or infinite.

A common characteristic of the above problems is that the open loop policy is easy to obtain, and decouples the dynamics of the queues. Further challenge is offered by the general routing problem in a general queueing network. Take for example the problem of routing a packet from node N_1 to node N_2. Suppose there are m alternative paths, each one of them involving an intermediate node. While the approach described above is applicable in principle, there is certain difficulty with step 3. Here, the differential costs for two queues in tandem are need. The methodology used of the appendix leads in this case to a recursion in two variables. Work is needed to find efficient solutions to it. Finally, the potential performance advantage of using several iterations of the value iteration algorithm has to be balanced against the extra complexity.

5. CONCLUSIONS

The main result of this paper is the derivation of an efficient recursion to compute the differential costs for a queue in isolation. This substantially extends Krishnan and Ott's[1] methodology to the control of queues, in that it allows the treatment of general cost structures. We have also recast the problem in the framework of open-loop feedback control policies and the value iteration algorithm. Future work should include:

(a) Derivation of efficient recursions for the differential cost for two queues in tandem. This would allow to extend the method beyond the queues in parallel paradigm.

(b) Quantification of the advantages of using further iterations of the value algorithm.

(c) Extension of the methodology to handle better initialization policies. Candidates include non-randomized splitting policies[15] [16] for the case of parallel queues.

APPENDIX

Derivation of the differential costs for a Markovian queue in isolation

The goal of this appendix is to derive recursions for the differential costs for a queue in isolation.

Let $c(i)$ denote the expected immediate cost when there are i customers in a queue system that receives Poisson arrivals at rate λ and has an exponential server with state dependent service rate $\mu(i)$. Note that no routing decisions are involved for this single queue system, and let ν denote the total transition rate that has been used to convert the continuous- time model into a discrete-time one. Then, the recursive equation that determines the cost-to-go $W^{\beta}(i)$, for an initial state is i, is given by:

$$(1-\beta)W^{\beta}(i) = c(i) + \beta\frac{\lambda}{\nu}[W^{\beta}(\lceil i+1\rceil^{L}) - W^{\beta}(i)] + \beta\frac{\mu(i)}{\nu}[W^{\beta}(\lfloor i-1\rfloor^{0}) - W^{\beta}(i)] \qquad \text{(A.1)}$$

where

$\lceil i+1\rceil^{L} \overset{\Delta}{=} \min(i+1, L)$, $\quad \lfloor i-1\rfloor^{0} \overset{\Delta}{=} \max(i-1,\ 0)$,

and L denotes the maximum possible number of customers in the system.

We are interested in

$$\Delta(i) \overset{\Delta}{=} \lim_{\beta\to 1}[W^{\beta}(i+1) - W^{\beta}(i)], \qquad \text{(A.2)}$$

with the boundary condition $\Delta(L) = 0$ imposed by the finiteness of the buffer.

Taking limits in (A.1) leads to the recursion

$$\Delta(L) = 0 \qquad \text{(A.3a)}$$

$$\Delta(i-1) = \left(\frac{\lambda}{\mu(i)}\right)\Delta(i) + \left(\frac{\mu(i)}{\nu}\right)^{-1}[c(i) - W^{av}],\ \ 0 < i \le L, \qquad \text{(A.3b)}$$

where

$$W^{av} = \Big[\sum_{i=0}^{L}\frac{\lambda^{i}}{\prod_{j=0}^{i}\mu(j)}\Big]^{-1}\Big\{\sum_{i=0}^{L}c(i)\frac{\lambda^{i}}{\prod_{j=0}^{i}\mu(j)}\Big\}, \qquad \text{(A.4)}$$

where by convention $\mu(0)=1$. The above expression assumes that $\sum_{i=1}^{L}\frac{\lambda^{i}}{\prod_{j=1}^{i}\mu(j)} \neq 1$. The equality case can easily be handled, though. The infinite buffer case demands the condition[1] $\sum_{i=1}^{L}\frac{\lambda^{i}}{\prod_{j=1}^{i}\mu(j)} < 1$, and the use of the recursion

$$\Delta(0) = \left(\frac{\lambda}{\nu}\right)^{-1}[W^{av} - c(0)] \qquad \text{(A.5a)}$$

$$\Delta(i+1) = \left(\frac{\mu(i)}{\lambda}\right)\Delta(i) - \left(\frac{\lambda}{\nu}\right)^{-1}[c(i+1) - W^{av}],\ i>0, \qquad \text{(A.5b)}$$

in lieu of (A.3).

---

[1] General conditions for the existence of the limit of the left hand side of (A.1) are studied by Lippman[17].

REFERENCES

1. K.R. Krishnan and T.J. Ott, "State-Dependent Routing for Telephone Traffic: Theory and Results," *Proceedings of 25th IEEE Conference on Decision and Control,* Athens, Greece, December 1986, pp. 2124-2128.

2. B. Hajek, "Optimal Control of Two Interacting Service Stations", *IEEE Trans. Autom. Contr., vol. AC-29, n. 6, pp.491-499, 1984.*

3. W. Lin and P.R. Kumar, "Optimal Control of a Queueing System with two Heterogeneous Servers," IEEE Trans. Autom. Contr., vol. AC-29, n. 8, August 1984, pp. 696-702.

4. S. Stidham, "Optimal Control of Admission to a Queueing System," *IEEE Trans. Autom. Control,* Vol. AC-30, n. 8, August 1985, pp. 705-713.

5. R.A. Milito and H. Levy, "Modeling and Dynamic Scheduling of a Queuing System with Blocking and Starvation", to be published in the December '89 issue of *IEEE Trans. Commun.*

6. K.R. Krishnan, "Joining the Right Queue: A Markov Decision-rule", *Proceedings of the 26th Conference on Decision and Control,* Los Angeles, CA, December 1987, pp. 19863, 1968.

7. K.R. Krishnan, "State-Dependent Allocation of Job Stream to Parallel Workstations," *Proceedings of the 27th Conference on Decision and Control,* Austin,TX, December 1988, pp. 2332,2333.

8. R. A. Milito and M. Cudjoe, "Dynamic Routing Policies for Multi-Class Traffic", 1st *AT&T Teletraffic Theory Symposium, May 1989.*

9. J.D.C. Little, "A Simple Proof of $L = \lambda W$," *Oper. Res., vol. 15, 1961, pp. 383-387.*

10. H.C. Tijms, *Stochastic Modeling and Analysis: A computational approach,* J. Wiley & Sons, 1986.

12. M. Aoki, *Optimization of Stochastic Systems,* Academic Press, 1967.

13. D.P. Bertsekas, *Dynamic Programming and Stochastic Control,* Academic Press, 1976.

14. S. Lippman, "Applying a new device in the optimization of exponential queueing systems," *Oper. Res. 23, 687-710, 1975.*

15. B. Hajek, "Extremal Splitting of Point Processes," *Mathematics of Operations Research,* vol 10, n.4, November 1985, pp. 543-556.

16. Z. Rosberg, "Deterministic Routing to Buffered Channels," *IEEE Trans. on Commun.*, vol. COM-34, n. 5, May 1986, pp. 504, 507.

17. S.A. Lippman, "Semi-Markov decision processes with unbounded reward", *Management Sci.*, vol. 19, pp. 717-731,1973.

Optimal Distributed Routing Algorithms for Datagram Communication Networks[1]

James R. Yee
Department of Electrical Engineering-Systems
University of Southern California

Abstract

We consider the problem of routing in a communication network with datagram service. Relationships among four optimality conditions are presented. Two distributed algorithms are derived. The main idea underlying the construction of these algorithms is the use of a vector of step sizes. This results in an algorithm with better descent properties than those in the literature.

1 Introduction

An effective method for assigning routes to packets is an important component in the efficient operation of a data network. The routing problem where the computations are performed at one central node is very well understood. Fratta, Gerla, Kleinrock [1973] developed the Flow Deviation Method which is the Frank-Wolfe method [1956] specifically tailored to solve a convex multicommodity flow problem. Cantor and Gerla [1974] developed the Extremal Flow Method for the routing problem. This method is intimately related to simplicial decomposition. Gafni and Bertsekas [1984] developed a Newton type method which they call the Gradient Projection Method. These centralized algorithms are particularly useful in solving the routing subproblem in the overall communication network design problem, i.e. specifying the topology, capacity assignment and routing.

In the operation of a data network, it is widely agreed that a distributed algorithm is better than a centralized one since a distributed algorithm is more survivable. Gallager [1977] formulated the routing problem as the optimization problem given in (2)-(5) which is specifically aimed at distributed computation. He developed the first distributed routing algorithm that converges to a global optimal solution. The main drawback of Gallager's algorithm is that in order to prove convergence of the algorithm to an optimal solution, the scale factor used was chosen to be proportional to N^{-6} where N is the number of nodes in the network. To address the difficulty of slow convergence, Bertsekas, Gafni and Gallager [1984] make use of second derivatives to develop a Newton type method. Their algorithm has the advantage of good convergence when the solution is sufficiently close to an optimal solution. However, as with Newton methods, convergence to an optimal solution from an initial feasible solution is not guaranteed. The principal result in this paper is a distributed algorithm that always converges to an optimal solution with a better convergence rate than Gallager's algorithm.

We will use the Gallager's formulation given in (2)-(5) and the Cantor and Gerla formulation of the problem which has been used to develop centralized algorithms. Let r_{ij} be the expected rate (in bits/sec) at which traffic enters the network at node i and destined for node j. Let $t_i(j)$ be the expected rate (in bits/sec) of traffic flow at node i destined for node j. The decision variables, $\phi_{ik}(j)$, represent the fraction of traffic at node i destined for node j that is routed over link $(i, k) \in L$. The variables $\phi_{ik}(j)$ also represent the probability

---

[1] Supported in part by a contract from AIRMICS and in part by Contract DAAL 03-88-K0059 from the Army Research Office

that a packet at node i destined to node j is routed to node k. Let f_{ik} be the expected rate (in bits/sec) over link $(i,k) \in L$. The function $D_{ik}(f_{ik})$ represents the expected number of packets awaiting transmission over link $(i,k) \in L$. It is assumed that $D_{ik}(f_{ik})$ is an increasing, convex and continuously differentiable function of f_{ik} such that $D_{ik}(f_{ik}) \rightarrow \infty$ as $f_{ik} \rightarrow C_{ik}$, where C_{ik} is the capacity of link (i,k). One such formula derived by Kleinrock [1964] under the assumption that each link can be modeled as an M/M/1 queue is

$$D_{ik}(f_{ik}) = \frac{f_{ik}}{C_{ik} - f_{ik}} \tag{1}$$

where C_{ik} is the capacity in bits/second of link (i,k).

The routing problem aimed at distributed computation first formulated by Gallager [1977] is

$$\min_{\phi \in \Phi} D(f) = \sum_{i,k} D_{ik}(f_{ik}) \tag{2}$$

subject to

$$\begin{aligned} t_i(j) - \sum_{\ell \in A(i)} t_\ell(j)\phi_{\ell i}(j) &= r_{ij} \quad \forall\, i,j \in V & (3)\\ f_{ik} - \sum_j t_i(j)\phi_{ik}(j) &= 0 \quad \forall\, (i,k) \in L & (4)\\ f_{ik},\ t_i(j) &\geq 0 \quad \forall\, (i,k) \in L;\ i,j \in V & (5) \end{aligned}$$

where

$$\Phi = \{\phi_{ik}(j) \geq 0 \mid \sum_k \phi_{ik}(j) = 1 \quad \forall i,j \in V,$$

and for each i,j there is a sequence of nodes $i,k,\ell,\ldots,m,j$ such that $\phi_{ik}(j) > 0, \phi_{k\ell}(j) > 0, \ldots, \phi_{mj}(j) > 0\}$.

We shall refer to the problem in (2)-(5) as G. Gallager proves that for a fixed set of requirements r and a fixed set of routing variables ϕ, there is a unique set of nodal flows $t_i(j)$ satisfying (3). Also, each $t_i(j)$ is a nonnegative and continuously differentiable function of r and ϕ. Then, from (4), each f_{ik} is unique and is a nonnegative, continuously differentiable function of r and ϕ. Let $J_{ik}(\phi) = D_{ik}\{f_{ik}[t(\phi),\phi]\}$ where $f_{ik}(t,\phi)$ represents (4) and $t(\phi)$ is the unique solution to (2). Then (2)-(5) may be expressed as the following equivalent nonlinear program

$$\min_{\phi \in \Phi} J(\phi) = \sum_{i,k} J_{ik}(\phi). \tag{6}$$

We next present the formulation given in Fratta, Gerla and Kleinrock [1973] and Cantor and Gerla [1974]. Let $f_{ik}(j)$ be the rate of flow of packets destined to node j on link (i,k).

$$\min D(f) = \sum_{i,k} D_{ik}(f_{ik}) \tag{7}$$

subject to

$$\begin{aligned} \sum_k f_{ik}(j) - \sum_\ell f_{\ell i}(j) &= r_{ij} \quad \forall\, i,j \in V & (8)\\ f_{ik} - \sum_j f_{ik}(j) &= 0 \quad \forall\, (i,k) \in L & (9)\\ f_{ik}(j) &\geq 0 \quad \forall\, (i,k) \in L;\ j \in V. & (10) \end{aligned}$$

We shall refer to (7)-(10) as (FGK). These two formulations are intimately related but not quite equivalent. Each feasible solution to (3)-(5) corresponds to a unique feasible solution to (8)-(10). The variables $f_{ik}(j)$ can be expressed in terms of the nodal flows and routing probabilities by

$$f_{ik}(j) = t_i(j)\phi_{ik}(j). \tag{11}$$

Multiplying (3) by $\phi_{ik}(j)$ and summing over k and then substituting $f_{ik}(j)$, (3) is transformed into (8). Conversely, a feasible solution to (8)-(10) does not correspond to a unique feasible solution to (3)-(5) when $t_i(j) = 0$.

In section 2, we show the relationship among four sets of optimality conditions. We point out that the routing problem (aimed at real-time computation of routing variables) is a nonconvex programming problem. In section 3, we derive two distributed algorithms. The key feature of these algorithms is the use of a vector of stepsizes rather than the usual scalar stepsize. In section 4, we discuss properties of the algorithms.

2 Optimality Conditions

In this section, we specify four sets of optimality conditions for the two formulations of the routing problem. Propositions 1 and 2 give the relationships among the four sets of conditions.

Most nonlinear programming techniques for problems with differentiable objectives functions are gradient-based methods. Typical nonlinear programming algorithms perform all computations on one computer. As pointed out earlier, it is essential that computations be done in a distributed manner in communications networks. The following equations can be used by each node i to compute, in a distributed manner, a subset of the elements of ∇J

$$\frac{\partial J}{\partial r_{ij}} = \sum_k \phi_{ik}(j)[D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}}] \tag{12}$$

$$\frac{\partial J}{\partial \phi_{ik}(j)} = t_i(j)[D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}}] \tag{13}$$

where

$$\frac{\partial J}{\partial r_{jj}} = 0 \text{ and } D'_{ik}(f_{ik}) = \frac{dD_{ik}(f_{ik})}{df_{ik}}.$$

Each node i can estimate $D'_{ik}(f_{ik})$ for each outgoing link (i,k). Either it can estimate it directly or it can estimate f_{ik} and use an appropriate formula derived from a queueing analysis, e.g. (1).

For each node i, let $A(i)$ and $B(i)$ be the set of downstream neighbors of i and the set of upstream neighbors of i, respectively. Node i receives $\frac{\partial J}{\partial r_{kj}}$ from each downstream neighbor $k \in B(i)$. Using its current routing assignment $\phi_{ik}(j)$, its estimate of $D'_{ik}(f_{ik})$ and (12), node i can calculate $\frac{\partial J}{\partial r_{ij}}$. After estimating $t_i(j)$ and then using (13), node i can calculate $\frac{\partial J}{\partial \phi_{ik}(j)}$. Node i then transmits $\frac{\partial J}{\partial r_{ij}}$ to each upstream neighbor $\ell \in A(i)$ so that each upstream node ℓ can calculate $\frac{\partial J}{\partial r_{\ell j}}$ and $\frac{\partial J}{\partial \phi_{\ell i}(j)}$. After repeated application of (12) and (13), each node will have calculated the elements of ∇J related to its routing variables. That is, each node $i \in V$ will have calculated $\frac{\partial J}{\partial \phi_{ik}(j)}$ for each $k \in B(i)$ and $j \in V$.

The Kuhn-Tucker conditions for (G) are

(i) $\phi \in \Phi$

(ii) there exist multipliers $\{u_{ij} \mid \forall i,j \in V\}$ such that

$$\begin{aligned} \frac{\partial J}{\partial \phi_{ik}(j)} &= u_{ij} \text{ if } \phi_{ik}(j) > 0 \\ &\geq u_{ij} \text{ if } \phi_{ik}(j) = 0 \end{aligned} \tag{14}$$

for all $(i,k) \in L$ and $j \in V$.

Unfortunately, the objective function in (6) is not a pseudoconvex function. [A function J is said to be pseudoconvex if $J(\phi^2) \geq J(\phi^1)$ whenever $\nabla J(\phi^1)^t(\phi^2 - \phi^1) \geq 0$ for each $\phi^1, \phi^2 \in \Phi$.] In Example 3.1 of Yee [1985], a routing assignment is given that satisfies the above Kuhn-Tucker conditions, but is not optimal. Since the set Φ is a convex set, $J(\phi)$ is not a pseudoconvex function (which also implies that $J(\phi)$ is not convex). Thus, (G) is not a convex programming problem. Due to space limitations, we have to exclude this example as well as some of the proofs of results that are discussed later in this paper.

Note that the constraints in (FGK) are linear and form a convex set. The objective function of (FGK) is convex. Thus, (FGK) is a convex programmming problem. In a sense, the transformation in (11) "convexifies" (G) into the convex programming problem (FGK). The advantage of (FGK) is that its corresponding Kuhn-Tucker conditions, given in (16) below, are both necessary and sufficient to identify an optimal solution. Whereas, the decision variables in (G) are natural for implementation. However, any solution to (FGK) can be transformed into a solution to (G) by the following. For each i and j where $\sum_k f_{ik}(j) > 0$, let

$$\phi_{ik}(j) = \frac{f_{ik}(j)}{\sum_k f_{ik}(j)} \qquad \forall\, j \in V \tag{15}$$

For each i and j where $\sum_k f_{ik}(j) = 0$, $\phi_{ik}(j) \geq 0$ can be chosen arbitrarily as long as $\sum_k \phi_{ik}(j) = 1$. In the next section, we will develop distributed algorithms based upon both formulations. Consider the following three different solution sets.

(i) Using the Kuhn-Tucker conditions for (FGK), a set of sufficient conditions for $\phi \in \Phi$ to be optimal are the existence of multipliers σ_{ij} satisfying

$$\begin{aligned} D'_{ik}(f_{ik}) + \sigma_{kj} &= \sigma_{ij} \qquad && \forall\, t_i(j)\phi_{ik}(j) > 0 \\ &\geq \sigma_{ij} && \forall\, t_i(j)\phi_{ik}(j) = 0. \end{aligned} \tag{16}$$

Let $\Omega_{KTF} = \{\phi \in \Phi \mid (16) \text{ are satisfied }\}$.

(ii) Gallager has proven that the following conditions are sufficient for $\phi \in \Phi$ to solve (G)

$$D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}} \geq \frac{\partial J}{\partial r_{ij}} \qquad \forall\, (i,k) \in L \text{ and } j \in V. \tag{17}$$

Let $\Omega_G = \{\phi \in \Phi \mid (17) \text{ are satisfied }\}$.

(iii) Let $\Omega_{KTG} = \{\phi \in \Phi \mid (14) \text{ are satisfied }\}$.

We relate the three sets of conditions in the following result.

Proposition 1.

(i) $\Omega_G \subset \Omega_{KTF} \subset \Omega_{KTG}$.

(ii) If $t_i(j) > 0$ for all $i, j \in V$, then $\Omega_G = \Omega_{KTF} = \Omega_{KTG}$.

(iii) Consider some $\phi \in \Omega_{KTF}$. Suppose for some i,j,k, $D'_{ik}(f_{ik}) + \sigma_{kj} > \sigma_{ij}$; $t_i(j) = 0$ and $\phi_{ik}(j) > 0$. Then, $\phi \in \Omega_{KTF} - \Omega_G$.

The proof of Proposition 1 is given in Yee [1985]. This result points out the relationships among the three sets of conditions specified for routing problem. The Gallager conditions are sufficient, but not necessary. The Kuhn-Tucker conditions in (16) are both necessary and sufficient, while the Kuhn-Tucker conditions in (14) are only necessary. Part (ii) shows that if $r_{ij} > 0$ for all $i, j \in V$, then one has the flexibility of choosing Ω_G, Ω_{KTF} or Ω_{KTO} as the solution set. This flexibility makes the task of proving convergence easier.

In the remainder of this section, we relate a fourth set of optimality conditions to the three already presented. In the fourth set of optimality conditions, we use the less restrictive requirement that the objective function be pseudoconvex at a point (rather than pseudoconvex over the entire feasible region). The function $J(\phi)$ is said to be pseudoconvex at the point $\phi \in \Phi$ if $\nabla J(\phi)^t(\phi^1 - \phi) \geq 0$ for all $\phi^1 \in \Phi$ implies that $J(\phi^1) \geq J(\phi)$. Let $\Omega_c = \{\phi \in \Phi \mid J \text{ is pseudoconvex at } \phi\}$. Let $\Omega_{KTS} = \Omega_{KTG} \cap \Omega_c$. Then any point $\phi \in \Omega_{KTS}$ is an optimal solution (see Bazaraa and Shetty [1979]). In the following, we prove that Ω_{KT} and Ω_{KTS} are equivalent.

Proposition 2. $\Omega_{KTS} = \Omega_{KTF}$.

Proof. First, consider any $\phi \in \Omega_{KTS}$. Then ϕ is an optimal solution and is contained in Ω_{KTF}, which are necessary conditions. Thus $\Omega_{KTS} \subseteq \Omega_{KTF}$.

Next, we prove that $\Omega_{KTF} \subseteq \Omega_{KTS}$. Consider any $\phi \in \Omega_{KTF} \subset \Omega_{KTG}$. Then ϕ is an optimal solution and $J(\phi^1) \geq J(\phi)$ for any $\phi^1 \in \Phi$. Thus, whenever $\nabla J(\phi)^t\ (\phi^1 - \phi) \geq 0, J(\phi^1) \geq J(\phi)$ showing that J is pseudoconvex at ϕ. □

3 Two Distributed Algorithms

In this section, we develop two distributed routing algorithms for computer networks with datagram service. Both algorithms can be classified as feasible direction methods. For this class of algorithms, a feasible direction of descent is determined. In this paper, we use the descent direction employed in the Frank-Wolfe method. However, other algorithms can be developed by using a different feasible direction of descent. The next step in a feasible direction method is to determine how far to move from the current solution along the feasible direction of descent, i.e. a scalar step size. Instead, we will use a vector of step sizes so that the step size can be determined using distributed computation. The first algorithm is developed for (G) where the feasible region is the space of routing fractions. The second algorithm is developed for (FGK) where the feasible region the flow space.

For the Algorithm 1, we assume that $r_{ij} > 0$ for all $i, j \in V$. From a practical point of view, it is reasonable to set any r_{ij} that is zero to some infinitesimally small $\varepsilon > 0$. As pointed out in Proposition 1, this simplifies the proof of its convergence to an optimal solution.

Suppose the current solution is $\phi \in \Phi$ and f is the corresponding flow vector. Also, suppose that each node i has an estimate of $D'_{ik}(f_{ik})$ for each $k \in B(i)$ and has received $\frac{\partial J(\phi)}{\partial r_{kj}}$ from each neighbor. Let $a_{ik}(j) = D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}}$ for all i, j, k. For each $i, j \in V$, let U_{ij} denote the subset of $B(i)$ consisting of the downstream neighbors of i for which the routing variables will be allowed to be increased. We will precisely define U_{ij} later. Consider the following linear program

$$\min_{\psi \in \Phi} \sum_{i,j,k} a_{ik}(j)\psi_{ik}(j). \tag{18}$$

subject to

$$\begin{aligned} \sum_{k \in U_{ij}} \psi_{ik}(j) &= 1 \quad \forall\, i, j \in V \\ \psi_{ik}(j) &\geq 0 \quad \forall\, (i,k) \in L,\ j \in V. \end{aligned}$$

Note that if $a_{ik}(j)$ in (18) is replaced by $\frac{\partial J}{\partial \phi_{ik}(j)} = t_i(j) a_{ik}(j)$ then (18) is the LP subproblem arising when applying the Frank-Wolfe method to (G). Also, the optimal solution $\{\phi^o_{ik}(j)\}$ to (18) is an optimal solution when the term $t_i(j)$ is included. The solution to (18) is trivial. For each i, j, let $k_o(i,j)$ be that neighbor $k \in U_{ij}$ which has the smallest $a_{ik}(j)$. In the case of ties, arbitrarily choose any one of those $k \in U_{ij}$ with the smallest $a_{ik}(j)$. Then,

$$\psi_{ik}(j) = \begin{cases} 1 & \text{if } k = k_o(i,j) \\ 0 & \text{if } k \neq k_o(i,j). \end{cases} \tag{19}$$

For all i, j, k, let $d_{ik}(j) = \psi_{ik}(j)\phi_{ik}(j)$. Then, the vector $d = [d_{ik}(j)]^t \in R^L \times R^N$ is a feasible direction of descent. We will use the superscript m to denote the iteration number. Since both ϕ and ψ are feasible, then for any scalar step size θ, $0 \leq \theta \leq 1$,

$$\begin{aligned} \phi^{m+1}_{ik}(j) &= (1-\theta)\,\phi^m_{ik}(j) + \theta\,\psi^m_{ik}(j) \\ &= \phi^m_{ik}(j) + \theta\,[\psi^m_{ik}(j) - \phi^m_{ik}(j)] \end{aligned} \tag{20}$$

is a feasible solution.

Usually, the next step is to choose the scalar step size θ to

$$\min_{0 \leq \theta \leq 1} J[\phi^m + \theta\, d^m]. \tag{21}$$

Solving the one dimensional optimization problem in (21) by the Fibonacci search procedure would require the exchange of too much information between nodes to evaluate the objective function. To avoid this excessive exchange of information, we (i) use a vector of step sizes, and (ii) use a quadratic approximation for $J(\theta)$ in (21).

Let θ_i be the step size for node i. Then, for each i, j, k (19) becomes

$$\phi^{m+1}_{ik}(j) = \phi^m_{ik}(j) + \theta_i\,[\psi^m_{ik}(j) - \phi^m_{ik}(j)]. \tag{22}$$

Note that if $0 \leq \theta_i \leq 1$ for each $i \in V$, then $\phi^{m+1} \in \Phi$.

Let θ be the N column vector of step sizes. We next develop a procedure for choosing the step sizes. Define the vector operation, $\theta \circ d$, such that the $(i,j,k)^{\underline{th}}$ component of $\theta \circ d$ is formed by multiplying the $i^{\underline{th}}$ component of θ by the $(i,j,k)^{\underline{th}}$ component of d. That is, $(\theta \circ d)_{i,j,k} = \theta_i d_{ik}(j)$. By Taylor's theorem,

$$\begin{aligned} J(\phi^m + \theta \circ d^m) &= J(\phi^m) + \sum_i \theta_i \sum_{j,k} \frac{\partial J(\phi^m)}{\partial \phi^m_{ik}(j)}\, d^m_{ik}(j) \\ &\quad + \frac{1}{2} \sum_{i,j,k} \theta_i\, d^m_{ik}(j) \sum_{\ell,p,q} \frac{\partial^2 J(w)}{\partial \phi^m_{ik}(j) \partial \phi^m_{\ell,p}(q)}\, \theta_\ell\, d^m_{\ell p}(q) \end{aligned} \tag{23}$$

where $w = \alpha\phi^m + (1-\alpha)[\phi^m + \theta \circ d^m]$ for some $\alpha \in (0,1)$. We assume that the second partial derivatives are bounded. Then, there exists a scalar M such that the third term on the right hand side of (23) is bounded above by $\frac{1}{2} M \sum_i \theta_i^2$. Form the function $G : R^N \to R$

$$G(\theta) = J(\phi^m) + \sum_i \theta_i \sum_{j,k} \frac{\partial J(\phi^m)}{\partial \phi^m_{ik}(j)}\, d^m_{ik}(j) + \frac{1}{2} M \sum_i \theta_i^2. \tag{24}$$

By construction, $J(\phi^m + \theta \circ d^m) \leq G(\theta)$. Also, note that

$$G(\theta)|_{\theta=0} = J(\phi^m) \tag{25}$$

and

$$\left.\frac{\partial G(\theta)}{\partial \theta_i}\right|_{\theta=0} = \sum_j t_i(j) \; [\min_{k \in U_{ij}} \{D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}}\} - \frac{\partial J}{\partial r_{ij}}] \leq 0. \tag{26}$$

Intuitively, whenever $\min_{k \in U_{ij}} \{D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}}\} < \frac{\partial J}{\partial r_{ij}}$ for some i and j, then there exists a nonzero θ such that $G(\theta) < J(\phi^m)$. Since $G(\theta)$ upper bounds $J(\phi^m + \theta \circ d^m)$, $J(\phi^m + \theta \circ d^m) < J(\phi^m)$. We naturally choose θ to minimize $G(\theta)$. That is, the step sizes $\{\theta_i\}$ are determined by solving the following simple convex program

$$\min \quad G(\theta) \quad \text{subject to} \quad 0 \leq \theta_i \leq 1, \quad \forall\, i = 1, 2, \ldots, N. \tag{27}$$

Since $M > 0$, $G(\theta)$ is a convex function. The solution to (27), which we denote by $\overline{\theta}_i$, is

$$\overline{\theta}_i = \min \left\{ 1, \; \frac{1}{M} \sum_j t_i(j) \; [\frac{\partial J}{\partial r_{ij}} - \min_{k \in U_{ij}} \; \{D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}}\}] \right\}. \tag{28}$$

To complete the description of Algorithm 1, we must define the sets $\{U_{ij}\}$. Define the routing variable $\phi_{ik}(j)$ to be improper if $\phi_{ik}(j) > 0$ and $\frac{\partial J}{\partial r_{ij}} \leq \frac{\partial J}{\partial r_{kj}}$. A node k is said to be blocked with respect to destination j, if k has a routing path to j containing some link (ℓ, m) where $\phi_{\ell m}(j)$ is improper and $\overline{\theta}_\ell < 1$. Let $\overline{U}_{ij}$ be the set of nodes $k \in B(i)$ for which $\phi_{ik}(j) = 0$ and k is blocked with respect to j. Let $U_{ij} = B(i) - \overline{U}_{ij}$.

The description of the Algorithm $A(\phi)$ is as follows. For each destination j, node i keeps repeating the following steps.

1. Each node i estimates $D'_{ik}(f_{ik})$ for each $k \in B(i)$ and $t_i(j)$ for each destination j.
2. Node i receives $\frac{\partial J}{\partial r_{kj}}$ from each downstream neighbor k. Node i then calculates $\frac{\partial J}{\partial r_{ij}}$ using (12) and broadcasts this quantity to its upstream neighbors.
3. Node i determines the sets U_{ij} and $K_{ij} = \{k \in U_{ij} \mid D'_{ik}(f_{ik}) + \frac{\partial J}{\partial r_{kj}} \leq D'_{im}(f_{im}) + \frac{\partial J}{\partial r_{mj}}$ for all $m \in B(i)\}$. Then, node i determines $\{\psi_{ik}(j)\}$ using (20) where k_o is any node in the set K_{ij}.
4. Using (28), node i calculates $\overline{\theta}_i$. Then, the new set of routing variables are determined using (22).

Note that since the set K_{ij} may contain more than one element, A_ϕ is a point to set mapping.

The same ideas are employed in development of Algorithm 2 except that the control variables are the link flows rather than routing fractions. We shall refer to Algorithm 2 as $A(f)$. In the remainder of this section, we relax the assumption that each $r_{ij} > 0$. However, we will make the additional assumption that each $D''_{ik}(f_{ik})$ is a monotone increasing function. Note that this assumption is valid when each queue is modeled as an $M/M/1$ queue, e.g. (1).

Let $\mathcal{F}$ be the set of link flows that satisfies (8)-(10). A feasible direction of descent is found by solving the following linear program

$$\min_{g \in \mathcal{F}} \sum_{i,j,k} D'_{ik}(f_{ik}) g_{ik}(j). \tag{29}$$

subject to

$$g_{ik}(j) \;\; = \;\; 0 \;\; \forall\, i, j \in V, \; k \notin U_{ij}$$

where U_{ij} is the set of downstream neighbors which is not blocked.
The solution to (29) is given by a slight modification of the distributed Bellman-Ford algorithm. For each destination j, each node i receives σ_{kj} from each neighbor k and calculates σ_{ij} by

$$\sigma_{ij} = \min_{k \in U_{ij}} \left\{ D'_{ik}(f_{ik}) + \sigma_{kj} \right\}. \tag{30}$$

For each O-D pair i and j, the origin node i routes r_{ij} units of flow to the downstream neighbor, say $k_0(i,j)$, that achieves the minimization in (30). Node $k_0(i,j)$ in turn routes the received flow from i to its downstream neighbor which achieves the minimization in (30). This is repeated until the r_{ij} units of flow are routed on a shortest path from i to j. Let $g_{ik}(j)$ be the resulting flow over link (i,k) destined to node j. The $\{g_{ik}(j)\}$ solves (29). A feasible direction of descent for (FGK) is the vector composed of the components

$$d_{ik}(j) = g_{ik}(j) - f_{ik}(j) \qquad \forall\ (i,k) \in L;\ j \in V. \tag{31}$$

As in $A(\phi)$, we use a vector of step sizes rather than the usual scalar step size. However, the network entity (e.g. node, link) to associate a particular step size must be chosen judiciously. Consider the linear constraints of (G) in canonical form $Ax = b$. Note that the constraint matrix A in (G) is a block diagonal matrix consisting of N blocks where the i^{th} block constrains only the routing fractions at node i, $\{\phi_{ik}(j) \mid \forall\ k \in B(i),\ j \in V\}$. Thus, a convex combination (with weights θ_i and $1 - \theta_i$) of two sets of routing fractions, which are feasible with respect to the i^{th} block, is feasible. Alternatively, the constraint matrix could also be viewed as having $N(N-1)$ blocks (one for each O-D pair) and the algorithm could have been developed using step sizes $\{\theta_{ij}\}$. In the constraints of (FGK), there are N blocks (one for each destination). Thus, we associate the step size θ_j with those flows destined to node j. The update rule is

$$f_{ik}^{m+1}(j) = f_{ik}^{m}(j) + \theta_j\ [g_{ik}^{m}(j) - f_{ik}^{m}(j)] \qquad \forall\ (i,k) \in L;\ j \in V. \tag{32}$$

The step sizes are determined by upper bounding $D(f^m + \theta \circ d^m)$ by a quadratic separable convex function. That is,

$$\begin{aligned} D(f^m + \theta \circ d^m) &= D(f^m) + \sum_j \theta_j \sum_{i,k} D'_{ik} d_{ik}^m(j) \\ &\quad + \frac{1}{2} \sum_{i,j,k} \theta_j\ d_{ik}^m(j) \sum_{\ell,p} \frac{\partial^2 D(w)}{\partial f_{ik}^m(j) \partial f_{\ell,p}^m(j)}\ \theta_\ell\ d_{\ell p}^m(j) \end{aligned} \tag{33}$$

$$D(f^m + \theta \circ d^m) = D(f^m) + \sum_j \theta_j \sum_{i,k} D'_{ik}\ d_{ik}^m(j) + \frac{1}{2} \sum_{j,i,k} \theta_j^2\ D''_{ik}(w)\ [d_{ik}^m(j)]^2 \tag{34}$$

where $w = \alpha f^m + (1-\alpha)[f^m + \theta \circ d^m]$ for some $\alpha \in (0,1)$.
Let $M_f = \max_{i,k}\ \max_{g \in \mathcal{F}} D''_{ik}(g_{ik})$. Then,

$$D(f^m + \theta \circ d^m) \leq D(f^m) + \sum_j \theta_j \sum_{i,k} D'_{ik} d_{ik}^m(j) + \frac{1}{2} \sum_{i,j,k} \theta_j^2\ M_f\ [d_{ik}^m(j)]^2 \tag{35}$$

The right hand side of (35) is a separable convex function which is minimized by

$$\overline{\theta_j} = \min \left\{ 1,\ -\frac{1}{M_f} \frac{\sum_{i,k} D'_{ik} d_{ikj}}{\sum_{i,k} d_{ikj}^2} \right\}. \tag{36}$$

The determination of θ_j can be determined using distributed computation with the aid of an arborescence rooted at j. An arborescence rooted toward j is a directed spanning tree where there is exactly one arc directed away from each node, except j, and there is a unique path from every node to node j. The evaluation of (36) can be determined by node j by the following. Each node $i \neq j$ evaluates $\lambda_i = \sum_k D'_{ik} d_{ik}(j)$ and $\gamma_i = \sum_k d^2_{ik}(j)$. Each leaf node, say p, in the arborescence begins by sending $\Lambda_p = \lambda_p$ and $\Gamma_p = \gamma_p$ to its downstream neighbor in the arborescence. Each node then waits until it receives the message containing Λ_q and Γ_q from every upstream node q in the arborescence. The node,say k, then adds λ_k (γ_k) to the sum of the received Λ's (Γ's) and transmits the results to its downstream neighbor in the arborescence. Eventually, node j determines θ_j using (36).

An arborescence rooted away from node j is a directed spanning tree where there is exactly one arc directed toward each node, except j, and there is a unique path from node j to every node. An arborescence rooted away from node j can be used to broadcast the step size θ_j to every node. Each node can then update its routing variables using (32). Humblet [1983] developed a distributed algorithm to determine N (one for each destination) minimum weight arborescences in $O(N^3)$.

Define the routing variable $f_{ik}(j)$ to be improper if $f_{ik}(j) > 0$ and $\sigma_{ij} \leq \sigma_{kj}$. A node k is said to be blocked with respect to destination j, if k has a routing path to j containing some link (ℓ,m) where $f_{\ell m}(j)$ is improper and $\overline{\theta}_\ell < 1$. For Algorithm $A(f)$, let $\overline{U}_{ij}$ be the set of nodes $k \in B(i)$ for which $f_{ik}(j) = 0$ and k is blocked with respect to j. Let $U_{ij} = B(i) - \overline{U}_{ij}$.

Each iteration of Algorithm $A(f)$ consists of the following steps. It is assumed that a distributed algorithm such as the one in Humblet [1983] has been used to determine arborescences rooted toward and away from each destination. For each destination j, each node i does the following:

1. Each node i estimates $D'_{ik}(f_{ik})$ for each $k \in B(i)$.
2. Node i receives σ_{kj} from each downstream neighbor k. Node i then calculates σ_{ij} using (30) and broadcasts this quantity to its upstream neighbors.
3. Node i calculates $g_{ik}(j)$ for each neighbor k.
4. Using (36) and the arborescences, destination node j calculates $\overline{\theta}_j$ and broadcasts it to every other node.
5. Then, each node calculates the new set of routing variables are using (32).

The rate of descent in each iteration of $A(f)$ can be determined from (35) and (36). There are three cases: (i) $\overline{\theta}_j = 0$, (ii) $0 < \overline{\theta}_j < 1$ and (iii) $\overline{\theta}_j = 1$. When $\overline{\theta}_j = 0$, the solution is optimal and does not change. When $0 < \overline{\theta}_j < 1$, (35) becomes

$$D(f^m + \overline{\theta} \circ d^m) \leq D(f^m) - \frac{1}{2M_f} \frac{\{\sum_{i,k} D'_{ik} d^m_{ik}(j)\}^2}{\sum_{i,k} [d^m_{ik}(j)]^2}. \tag{37}$$

When $\overline{\theta}_j = 1$, then

$$D(f^m + \overline{\theta} \circ d^m) \leq D(f^m) + \frac{1}{2} \sum_{i,k} D'_{ik} d^m_{ik}(j) \tag{38}$$

where the second term on the right hand side is negative when the solution is not optimal. Note that the rate of descent of $A(f)$ compares favorably with the algorithm in Segall [1979] where the rate of descent is of the order $(M_f\ N^5)^{-1}$. In comparison, when N (the number of nodes) is large, the rate of descent of $A(f)$ is significantly better.

4 Properties of the Algorithms

When developing optimal distributed routing algorithms, it is customary to prove that the algorithm (i) determines loop-free routes and (ii) converges to a global optimal solution. Algorithms $A(\phi)$ and $A(f)$ possess these properties. Due to the space limitation, we omit the proofs here and refer the reader to Yee [1985].

Theorem 1. Suppose the marginal link delays $D'_{ik}(f_{ik})$ are positive. Given an initial loop-free routing assignment, then each iteration of algorithms $A(\phi)$ and $A(f)$ result in loop-free routing assignments.

Theorem 2. Let ϕ^0 be any initial feasible solution to G. Let $\phi^m \in A(\phi^{m-1})$ for all m. Then, $\lim_{m\to\infty} \phi^m \in \Omega_G$.

Theorem 3. Let f^0 be any initial feasible solution to FGK. Let $f^m \in A(f^{m-1})$ for all m. Then, $\lim_{m\to\infty} f^m \in \Omega_{KTF}$.

Acknowledgment: I am grateful to Professors John M. Wozencraft and Robert G. Gallager for their helpful comments on an earlier version of this work.

References

1. Bertsekas, D.P., E.M. Gafni and R.G. Gallager (1984), "Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks," *IEEE Trans. on Comm.*, COM-32, pp 911-919.

2. Cantor, D.G. and M. Gerla (1974), "Optimal Routing in a Packet-Switched Computer Network," *IEEE Trans. on Computers*, C-23, pp 1062-1068.

3. Frank, M. and P. Wolfe (1956), "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly*, Vol. 3, pp 95-110.

4. Fratta, L., M. Gerla and L. Kleinrock (1973), "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," *Networks*, Vol. 3, pp. 97-133.

5. Gafni, E.M. and D.P. Bertsekas (1984), "Two-Metric Projection Methods for Constrained Optimization," *SIAM J. Contr. Optim.*, Vol. 22, pp 936-964.

6. Gallager, R.G. (1977), "A Minimum Delay Routing Algorithm Using Distributed Computation," *IEEE Trans. on Comm.*, COM-25, pp 73-85.

7. Humblet, P.A. (1983), "A Distributed Algorithm for Minimum Weight Directed Spanning Trees," *IEEE Trans. on Comm.*, COM-31, pp 756-762.

8. Kleinrock, L. (1962), *Communication Nets: Stochastic Message Flow and Delay*, Ph.D. Thesis, Dept of EE, M.I.T.

9. Segall, A. (1979), "Optimal Distributed Routing for Virtual Line Switched Data Networks," *IEEE Trans. on Comm.*, COM-27, pp 201-209.

10. Yee, J.R. (1985), *Distributed Routing and Flow Control Algorithms for Communication Networks*, Ph.D. Thesis, Dept of EECS, M.I.T.

Stochastic Approximation Schemes for a Load Balancing Problem [1]

RAHUL SIMHA & JAMES F. KUROSE
Computer and Information Sciences Department, University of Massachusetts
Amherst, Mass. 01003

Abstract: We consider the optimization of performance measures defined for a queueing system with respect to a given set of parameters. We focus on the use of stochastic gradient algorithms and, in particular, we study the application of these algorithms to the sample problem of adjusting flow rates to queues and examine the case that on-line and typically noisy sensitivity measurements are made in the system. Two stochastic approximation schemes, one scheme from the literature and the other introduced here, are used to augment an optimization algorithm in order to realize an optimal set of parameters. The second scheme, sampling-controlled iteration, provides a method of dealing with biasedness in estimators, a property present in several relevant estimation procedures.

1. Introduction

The problem of determining optimal values for parameters that govern the behavior of queueing networks has recently received much attention in the literature [7,9,10,18,21,22,26,31], particularly so in replacing Monte-Carlo methods with direct stochastic approximation [9,22]. In this paper, we present a new stochastic approximation technique that is especially suited to the optimization of queueing systems. To develop our methods and compare them with the standard stochastic appproximation technique we focus on a particular application, the problem of adjusting flow rates to queues, which, when the performance measure is average delay, has also been widely studied as the problem of static load balancing [5,23,30] and also as a component of a routing problem [3,8].

Our contribution is to present a gradient-based optimization algorithm and to augment it into a stochastic approximation (optimization) procedure through the use of two schemes for interleaving an iterative optimization algorithm with the process of gradient estimation. One scheme is well-established and is commonly called stochastic approximation, referred to here as *stepsize-controlled iteration*; we also introduce an alternative iteration technique which we call *sampling-controlled iteration*, that has the added advantage of removing bias from estimators, the importance of this being that several commonly used estimators associated with queueing systems are known to be biased [17,29]. A similar scheme has been developed independently in [31] although we note that in [31], a nonstandard definition of convergence was used.

One of the main results of this paper is that we establish the theoretical convergence of the resulting algorithms. We present simulation results that compare the convergence behavior of the two schemes and, furthermore, also consider the situation in which feasibility constraints (that the total arrival rate to a queue should be less than the service rate) are unknown and must themselves be estimated from the system.

Problem definition: Consider the system in figure 1 in which arriving jobs are routed to the i^{th} queue with probability x_i where they receive service in a first-come-first-served fashion. The interarrival times of the jobs are independent random samples from an unknown distribution A with mean $\frac{1}{\lambda}$. Similarly, the service times at queue i are taken independently from an unknown distribution B_i with mean $\frac{1}{\mu_i}$. The set of parameters, such as λ and μ_i is denoted by Θ. In simulation systems [21,22] the arrival and service rates may be known whereas in other systems, these parameters may be have to be estimated on-line The performance metric we consider in this paper is the expected system time of a job We assume the existence of a differentiable

---

[1] This work was supported in part by the Office of Naval Research under contract number N00014-87-K-304

mean delay function $D(x_1, ..., x_K, \Theta)$ that is convex in $x_1, ..., x_K$ for every set of parameters Θ. We are now in a position to formally state the problem:

$$\min_{x_1,\dots,x_K} \quad D(x_1, ...x_K, \Theta) \qquad ..... \quad (P_1)$$

such that $\sum_{i=1}^{K} x_i = 1$, $\forall i : \; x_i \geq 0$ and $\forall i : \; \lambda x_i < \mu_i$. Note that the methods presented in this paper are not bound to this choice of a performance metric. Provided that appropriate estimators are used, the algorithms and theoretical results shown here apply to other parameters and metrics.

2. Approaches to Optimization

Various approaches to solving problem P_1 are discussed in [26]; these are categorized into methods that use analytic formulas and the method of direct stochastic approximation. Let $\hat{\Theta}$ denote an estimate of Θ. In the first approach, a queueing model is used to obtain an expression for the metric $D(X, \Theta)$. After a period of estimation, the function $D(X, \hat{\Theta})$ is minimized using either direct analytic methods [5,30] or numerical procedures [19]. The disadvantage with this approach is that if the estimates are not accurate, we might not obtain a solution close to the optimal policy or, even worse, an infeasible solution. This problem can be overcome if a numerical optimization algorithm that uses analytic formulas is *interleaved* with estimation: each optimization iteration uses an estimate of Θ constructed from all measurements observed since the beginning [4]. We will be interested in the second approach, in which modeling assumptions made about the distributions A and B_i are not valid or that, even if they are, the corresponding queueing problem is not amenable to analytic solution. We restrict our attention to stochastic approximation methods and make use of recent results in gradient estimation for queueing systems [13,24,28,29].

Stochastic approximation [25] refers to a technique based on strong theoretical foundations in which noise in input data is systematically masked out in the process of iterating towards a solution. This procedure has found several applications [20,32], including in areas related to computer science [7,11,15,18,21,31]. The load balancing problem has been studied in [18] where the distributions A and B_i are all taken to be exponential and the minimization of the mean delay function, D, is reformulated into a problem of minimizing weighted least squares of queue idle times. An idle time estimator is introduced and convergence to the optimum is demonstrated. In contrast, we assume very little about the nature of the arrival and service distributions and use gradient estimators. We note that several research efforts have recently been undertaken in the use of direct gradient estimation for stochastic approximation [7,9,10,21,22,31]. Our work differs from [7,9,10,21,22] in that we present a new stochastic approximation technique that may be applied to any stochastic gradient algorithm and, more importantly, this technique handles bias in estimators, a problem prevalent with estimators for queueing systems [17,29]. A similar technique has been independently proposed by [31], although in [31], stronger conditions are imposed and a nonstandard definition of convergence is used.

3. Two Stochastic Approximation Schemes

It is assumed here that some gradient estimation technique is available and, in this paper, details concerning gradient estimation are not discussed. The reader is referred to the vast literature on this subject (for example, [3,13,24,28,29]). We introduce now some of the notation used in describing the computation at each step (t denotes time and n, the iteration number of the algorithm):

τ_n - the random time at which the n^{th} iteration of the algorithm starts execution.

$M_i^{(n)}$ - the (random) number of samples obtained of $\frac{\partial D}{\partial x_i}$ since τ_{n-1}, i.e., since the $(n-1)^{th}$ iteration. We assume that each estimation of the derivative is performed over a busy period for that queue and that the measurement obtained from a single busy period constitutes a sample. As we will see, in the case of sampling controlled iteration, the division of samples into busy periods is unnecessary.

$L^{(n)}$ - a sequence of positive integers to be described later.

$\hat{f}^{(n)}_{iL^{(n)}}(x_i)$ - an estimate of $\frac{\partial D}{\partial x_i}$ constructed from $M_i^{(n)}$ samples of $\frac{\partial D}{\partial x_i}$ since τ_{n-1}. For convenience of description, we use the notation $f_i(x_i)$ to denote the partial derivative, $\frac{\partial D}{\partial x_i}$.

Given the above definitions, we can now more formally define τ_n as $\tau_n = \inf_{t>\tau_{n-1}}\{\forall i : M_i^{(n)} \geq L^{(n)}\}$. Thus, τ_n is the time at which at least $L^{(n)}$ samples have been obtained from each queue since the last iteration.

The optimization algorithm below is based on general gradient projection methods and differs from the usual projection method in the manner by which inequality constraints are handled. First presented in a microeconomic context [12], the algorithm has since been used in the area of general resource allocation by [14], with mappings that permit decentralized execution in different communication structures. For handling inequality constraints, the construction of the feasibility set, $A^{(n)}$, in [12] was corrected for in [19] (described below) and the use of second derivative information and constant-order communication was considered. We also note that related gradient-based approaches have been used in several routing and load balancing problems [3,8]

The computation at each step is described by the equation: $X^{(n+1)} = X^{(n)} + a_n \Delta X^{(n)}$, where a_n is a scaling factor or *stepsize* and $\Delta x_i^{(n)}$, the i^{th} component of $\Delta X^{(n)}$, is given by

$$\Delta x_i^{(n)} = -\left(\hat{f}^{(n)}_{iL^{(n)}} - \frac{1}{|A^{(n)}|}\sum_{j\in A^{(n)}} \hat{f}^{(n)}_{jL^{(n)}}\right) \tag{1}$$

Here, $\Delta x_i^{(n)} = 0, \forall i \notin A^{(n)}$ and $A^{(n)}$ is the set described below:

```
/* Algorithm for computing the set $A^{(n)}$ */
(i)    For all $i$, sort $\hat{f}^{(n)}_{iL^{(n)}}$ and set $E = \{m | \hat{f}^{(n)}_{mL^{(n)}} = \min_i \hat{f}^{(n)}_{iL^{(n)}}\}$.
(ii)   Do step (iii) for each $j, j \notin E$ in ascending order of $\hat{f}^{(n)}_{jL^{(n)}}$.
(iii)  If $x_i^{(n)} + a_n \Delta x_i^{(n)} \geq q_{min}$ as a result of the reallocation defined by equation 1 above with
       $A^{(n)} = E \cup \{j\}$, then set $E = E \cup j$. The use of $q_{min}$ is discussed below.
(iv)   Set $A^{(n)} = E$
```

The following intuitive explanation of equation (1) may be given. The probability of sending a job to a particular queue i is increased ($\Delta x_i > 0$) if the estimated *marginal* expected completion time is less than that of the estimated average (over all queues) of the marginal expected completion time. In contrast, if this estimated marginal completion time for queue i is higher than the measured average, then the probability of sending a job to that queue is decreased ($\Delta x_i < 0$). The set $A^{(n)}$ is used to maintain the feasibility constraints, $x_i^{(n)} \geq 0$. Note that if $x_i^{(n)} = 0$ then no jobs would be sent to that queue and hence for $m > n$ we would not obtain any samples from the i^{th} queue. To ensure that $x_i^{(n)}$ remains positive, we thus force $x_i^{(n)} \geq q_{min} > 0$ and add this as a constraint to problem P_1.

Let X^* be the solution to the problem P_1. Due to the noise in the estimates, $\hat{f}^{(n)}_{iL^{(n)}}$, the update algorithm described above would not ordinarily converge to the optimum policy, X^*, unless some care is taken to mask out the effects of the noise. We now present two such methods and thereby, describe how a basic optimization algorithm is augmented into a stochastic approximation procedure. The first of these schemes is based on the standard stochastic approximation technique [20,25] and we call this *stepsize-controlled iteration.* The second technique, introduced here, is called *sampling-controlled iteration.* The two schemes are presented implicitly by stating the assumptions required for convergence. Thus, the first scheme is captured in statement $A1$ of the assumptions for Theorem 1 whereas the second one is described in statement $A4$ below.

Stepsize-controlled iteration:

$A1$. The stepsizes satisfy the following conditions: $\forall n : a_n > 0$, $\lim_{n\to\infty} a_n = 0$ and $\lim_{n\to\infty} \sum_{i=1}^{n} a_i = \infty$. Furthermore, $\forall n : L^{(n)} = L$ where $L \geq 1$.

$A2$. We assume that a strong law of large numbers exists for the random variables $\Delta x_i^{(n)}$ in the sense that if $\forall n : E[\Delta x_i^{(n)}] > \delta$ for some $\delta > 0$ then $\lim_{n\to\infty} \sum_{j=1}^{n} \Delta x_i^{(j)} = \infty$ *a.s.*. Note that if the queues have a regenerative property when the queue is empty then successive random samples of $\Delta x_i^{(n)}$ are independent samples and thus, we may apply the standard form of the strong law.

$A3$. The estimates $\hat{f}_{iL^{(n)}}^{(n)}$ are independent, unbiased estimates of $\frac{\partial D}{\partial x_i}$ with finite variance.

THEOREM 1: Under assumptions stated in the problem definition and $A1 - A3$, we have $\lim_{n\to\infty} X^{(n)} = X^*$ *a.s.*

PROOF: The details of the proof are presented in [26]. We note, however, that proofs for general gradient-based optimization algorithms are presented in [20] and modification is required here only to handle the feasibility set, $A^{(n)}$.

Sampling-controlled iteration:

$A4$. The sequence of integers $L^{(n)}$ satisfies the following conditions: $L^{(0)} \geq 1$ and $\forall n : L^{(n+1)} > L^{(n)}$. In addition, $\forall n : a_n = a$ where $a > 0$.

$A5$. We assume that the estimates $\hat{f}_{iL^{(n)}}^{(n)}$ are strongly consistent in the following sense:

$$\lim_{n\to\infty} \hat{f}_{iL^{(n)}}^{(n)}(x_i) = \frac{\partial D}{\partial x_i} \quad a.s.$$

$A6$. We assume that the deterministic optimization algorithm of equation (1) is stable under small perturbations, i.e., given $\delta > 0$ there exists $N < \infty$ and $\epsilon > 0$ such that if $\frac{\partial D}{\partial x_i} + \epsilon_n$ is used instead of $\frac{\partial D}{\partial x_i}$ in equation (1) and if $\sup_n |\epsilon_n| < \epsilon$, then for all X and $N \leq n \leq 2N$, $\left|X^{(n)} - X^*\right| \leq \delta$. We note that if the above condition were not true for a deterministic algorithm, then it would mean that there exist perturbations arbitrarily small which cause the algorithm to deviate by large amounts [6].

$A6'$. We assume that for each x_i and each sample path, there exists M and an open neighbourhood of x_i, denoted by $N_\epsilon(x_i)$, such that $\forall n > M$ and $\forall x \in N_\epsilon(x_i)$, $\left|\hat{f}_{iL^{(n)}}^{(n)}(x) - f_i(x)\right| < \epsilon$. In [26], it is demonstrated that this uniformity assumption is not overly restrictive by showing that additive white noise satisfies this condition.

THEOREM 2: Under the conditions described in earlier sections and $A4 - A6$ or $A4, A5, A6'$, there exists a stepsize $a > 0$ such that $\lim_{n\to\infty} X^{(n)} = X^*$ *a.s.*

PROOF: A proof using $A4 - A6$ follows from Theorem 3 in [6], in which large deviation estimates are constructed using the strong consistency of the gradient estimators. These provide large deviation bounds for the process $X^{(n)}$, from which convergence is proved. A proof based on $A4, A5, A6'$ may be found in [26].

A few comments are in order here. First, most of the known estimators associated with queueing systems such as the derivative estimators [24,29]and the estimator in [18] are known to be biased [17]. The bias is usually due to the nonlinearity present in the estimator formulas and also to the correlation present in a single cycle of a regenerative system [17]. In these cases, although assumption $A3$ is not satisfied when using stepsize-controlled iteration, there will still be convergence to a stable, albeit possibly suboptimal policy. In this case one may hope that the bias is small enough not to matter or that there is some cancelling effect due to symmetry (such as when the service distributions are identical). Secondly, sampling-controlled iteration does not require that the estimators be unbiased, only that they be strongly consistent. Strong consistency has been established in several cases for likelihood estimators [24], the infinitesimal perturbation analysis estimator [28] and formula-based estimators, such as the one in equation (2) of this paper.

Thirdly, we do not know whether assumption $A6'$ is satisfied by any of the known estimators and instead, make the following intuitive appeal. The arrival and service distributions may be assumed independent of the routing policy and therefore, the degree of noise present along a particular sample path for fixed X should remain independent of X. Thus, for that sample path, if we change x_i to $x_i + \delta$, then the rate at which noise is filtered out in estimating $f_i(x_i)$ should be approximately the same as that for $f_i(x_i + \delta)$ for small enough δ. Finally, a third algorithmic scheme results from taking both decreasing stepsizes and an increasing number of samples, although convergence in this case will be quite slow. For this case assumptions $A1 - A5$ are sufficient to prove convergence and hence a standard convergence result may be derived in place of the result in [31].

4. Experimental Results and Discussion

In this section we experimentally examine several aspects of the the convergence of the algorithms here. For experimental results in this paper, we use the Finite Perturbation Analysis estimator [3] instead of Infinitesimal Perturbation Analysis [29]. Theoretically, Finite Perturbation Analysis, is only a heuristic from which an estimate can only be made arbitrarily close but not equal to the true derivative. However, we find that it is simple to implement, and, in our simulation results, appears to be the best estimator among those studied here for the load balancing problem.

4.1 Comparison of the Two Schemes

For our simulation results, we considered a homogeneous system of six queues and took the interarrival and service distributions to be exponential with parameters $\lambda = \mu_1 =, ..., \mu_K = 2$ (Thus, $x_1^* =, ..., x_K^* = \frac{1}{6}$ and $D(X^*, \Theta) = 0.6$). The initial vector, $X^{(0)}$ was taken to be $X^{(0)} = 0.3, 0.2, 0.1, 0.1, 0.15, 0.15$. Next, we took the perturbation constant to be $\Delta r_i = 0.001$. Each point plotted in each curve is the result of an average taken from 50 simulation runs. The confidence intervals were found to be less than one-hundreth of each y-axis unit and are omitted from the graphs. Figure 2 shows a plot of $D(X^{(n)}, \Theta)$ at various values of n, i.e., a plot of the expected job delay for various policies generated through the progress of the algorithm. In order to study the progress made through time, we plot the performance metric after every 500 departures from the entire system. The two curves plotted in figure 2 may be characterized as follows:

- C_1: Stepsize-controlled iteration (standard stochastic approximation) with $a_n = 0.1/n$, $a_0 = 0.1$, $L^{(n)} = L = 50$. Although L is usually taken to be unity, we have observed that it is possible to obtain slightly better performance with higher values of L and, certainly, for a given choice of system parameters, there is likely to be an optimal value. A search for an optimal value of L is extremely time consuming and since our primary purpose was to make a comparison between sampling-controlled iteration and standard stochastic approximation, we settled on the chosen value of L.

- C_2: The second stochastic approximation scheme, sampling-controlled iteration, with $a_n = a = 0.01$, $L^{(n)} = L^{(n-1)} + 10$ and $L^{(0)} = 10$. As with the above scheme, the performance here also depends on the choice of a and the recurrence relation for $L^{(n)}$. We mention that the stepsize a was chosen after a brief search in the interval $[0.005, 0.1]$ and, due to the cost of searching for an optimal recurrence relation, the choice of 10 was entirely arbitrary.

We observe, from figure 2, that sampling-controlled iteration results in poor performance in the initial stages of the procedure and then rapidly gains on standard stochastic approximation, with clearly better performance towards the middle and later stages. The curves in figure 2 reveal an average trend but do not indicate how an arbitrary single run might perform in comparison with the mean performance. We attempt to capture this variability in performance in figure 3 which plots the variances of mean delay for the policies produced by the two stochastic approximation schemes:

- C_3: Stepsize-controlled iteration with $a_n = 0.1/n$, $a_0 = 0.1$ and $L = 50$.
- C_4: Sampling-controlled iteration with $a_n = a = 0.01$, $L^{(n)} = L^{(n-1)} + 10$ and $L^{(0)} = 10$.

We observe that the variance in policies produced by stepsize-controlled iteration is significanly higher than in the policies produced by sampling-controlled iteration.

4.2 Unknown Feasibility Constraints

We have thus far only treated the case in which the constraint $\lambda x_i^{(n)} < \mu_i$ is automatically satisfied when $x_i^{(n)} < 1$. However, it is possible that in certain systems, $\frac{\mu_i}{\lambda} < 1$ for some i and thus, ensuring that $x_i^{(n)} < 1$ does not guarantee that $\lambda x_i^{(n)} < \mu_i$. When $x_i^{(n)}$ is inadvertently made greater (through algorithmic iteration) than $\frac{\mu_i}{\lambda}$, the i^{th} queue gets overloaded and hence queue instability occurs. This problem may be overcome with modifications to both the optimization algorithms as well as the stochastic approximation procedure. We refer the reader to [27] for details and instead, present only a brief overview and a simulation result. Let $\mu_i^{(n)}$ be an estimate constructed from observing all service time samples from $t = 0$ up to τ_n, the time of the n^{th} iteration ($\lambda^{(n)}$ is similarly defined). Define $b_i^{(n)} = \frac{\mu_i^{(n)}}{\lambda^{(n)}}$, the current estimate of the upper constraint on x_i before the n^{th} iteration. The algorithm is modified so that at each iteration $0 < x_i^{(n)} < b_i^{(n)} - \zeta$ for some small $\zeta > 0$. If ζ is chosen so that $\forall i : x_i^* < \frac{\mu_i}{\lambda} - \zeta$ then it can be shown [27] that $\lim_{n\to\infty} X^{(n)} \to X^*$ *a.s.*.

Figure 4 shows a graph that compares stepsize and sampling-controlled iteration for the six-queue system described above with both iteration schemes modified as outlined above. In this case the service and arrival parameters were taken to be $\mu_i = 2$ and $\lambda = 8$ and thus we have the constraint $x_i < 0.25$. The initial policy was taken to be $X^{(0)} = (0.2, 0.2, 0.18, 0.18, 0.12, 0.12)$, which. although feasible in this case, need not have been a feasible starting solution [27].

- C_5: Stepsize-controlled iteration with $a_n = 0.1/n$, $a_0 = 0.1$ and $L = 50$.
- C_6: Sampling-controlled iteration with $a_n = a = 0.001$, $L^{(n)} = L^{(n-1)} + 10$ and $L^{(0)} = 10$.

We observe that in this case the algorithm under sampling-controlled iteration results in a better convergence profile.

4.3 Other Estimators

We consider the use of the likelihood-ratio estimator in [13,24]. Our experience with the simulated system described above shows that, in general, the perturbation analysis estimator results in better performance. For example, figure 5 shows the convergence profile for the following parameters and observe that, in this case, using the perturbation analysis estimator (curves C_1 and C_2) results in better performance:

- C_7: Sampling-controlled iteration with $a_n = a = 0.01$, $L^{(n)} = L^{(n-1)} + 10$ and $L^{(0)} = 10$ using a likelihood estimator [13,24].

Finally, for the special case of our simulation system, in which each queue is an $M/M/1$ queue, we consider the use of an estimator based on the expression for the derivative of the mean delay for this queue:

$$\hat{f}_i^{(n)}(x_i) = \frac{\hat{\mu}_i^{(n)}}{\left(\hat{\mu}_i^{(n)} - \hat{\lambda}^{(n)} x_i\right)^2} \tag{2}$$

where $\hat{\lambda}^{(n)}$ and $\hat{\mu}_i^{(n)}$ are described in section 5.2. Figure 6 shows the results in comparing the above estimator in equation (2) with the Perturbation Analysis estimator of previous sections. We plot the convergence profiles C_1 and C_2 together with the following curves:

- C_8: Stepsize-controlled iteration with $a_n = 0.1/n$, $a_0 = 0.1$ and $L^{(n)} = L = 50$ using the estimator in equation (2).
- C_9: Sampling-controlled iteration with $a_n = a = 0.05$, $L^{(n)} = L^{(n-1)} + 10$ and $L_0 = 10$ using the estimator in equation (2).

We observe that, although the estimator (2) uses information collected since $t = 0$ and appears to be the natural choice for an $M/M/1$-based system, the resulting performance is no better than when Perturbation Analysis is used (curves C_1 and C_2). We note that another formula-based estimator based on $M/G/1$-queues is described in [26].

4.4 Discussion

While we cannot state conclusively that sampling-controlled iteration is better than stepsize-controlled iteration, we seek to provide an intuitive explanation of its better performance. Consider the simple case in which the inequality constraints are not active at the optimum. We use the notation $d(X^{(n)})$ to denote the *spread* of the marginal delays: $d(X^{(n)}) = \max_{i,j} \left| f_i(x_i^{(n)}) - f_j(x_j^{(n)}) \right|$. Then, as the algorithm converges, $d(X^{(n)}) \rightarrow 0$, implying that the derivatives get closer and closer and thereby resulting in the convergence of the policy to a Kuhn-Tucker point. Now, each algorithmic step represented by equation (1) uses precisely the above spread of derivatives to achieve movement of the current policy in the direction of the optimum, and thus, as $d(X^{(n)})$ diminishes in magnitude, it becomes necessary to obtain more refined estimates of the derivatives and also to remove small sample bias in order to distinguish them within a ball of radius $d(X^{(n)})$. This, then, is the principle behind sampling-controlled iteration.

Similar observations have been made by other research efforts in stochastic approximation [2,16] as well as in other areas [1]. In [2], a fixed number of iterations was considered and the optimal selection of stepsizes and sample sizes was studied in order to minimize mean-squared-error for the particular case of the Robbins-Munro algorithm [25]. However, we caution that stepsize-controlled iteration is still well-suited to the problem examined here and may be the better scheme for other problems. In the early stages of our research, it was thought that the general principle above might be extended by computing confidence intervals about the estimates of the derivatives to determine the sampling period, $L^{(n)}$, so that the spread of derivatives could be distinguished with high probability. In this manner one would sample only as much as necessary. However, variance estimates required to estimate confidence intervals have a higher noise content than the derivative samples and, therefore, the use of confidence information resulted in poorer performance.

5. Conclusions

In this paper, we described an important parameter optimization problem. Our concerns focused on systems in which little is known or computable about the distributions involved and each associated parameter must itself be estimated. Two stochastic approximation schemes, the standard method as well as one introduced here, were presented to augment an optimization algorithm. The convergence of both these schemes was established and simulation results were presented that provided a basis for comparison. It was found that the second scheme, sampling-controlled iteration, in addition to its capability of removing biasedness in estimators, resulted in better performance than standard stochastic approximation. For future work, we observe that in the case that no estimators for the derivatives are available, Kiefer-Wolfowitz methods [20,32], which are often used in such situations, need to be examined more carefully. Finally, we note that, in a distributed system, it is possible to have several interacting controllers simultaneously updating their routing vectors and thus issues such as uniqueness of optimality and convergence must be addressed.

Acknowledgements: The authors would like to thank Prof. Don Towsley of the Department of Computer Science at the University of Massachusetts for several insightful comments and valuable discussions. We also would like to thank Prof. Chris Cassandras of the Department of Electrical Engineering and Prof. Paul Dupuis of the Department of Mathematics and Statistics at University of Massachusetts for their useful feedback regarding this paper.

REFERENCES

[1] A.G.Barto and M.I.Jordan, "Gradient Following Without Back-propagation in Layered Networks", *Proc. IEEE First Annual Conf. on Neural Networks*, San Diego, California, 1987.

[2] H.D.Block, "Estimates of Error for Two Modifications of the Robbins-Munro Stochastic Approximation Process", *Ann. Math. Stat.*, Vol. 28, pp. 1003-1010.

[3] C.Cassandras, V.Abidi and D.Towsley, "Distributed Routing with On-line Marginal Delay Estimation", *IEEE Trans. Communications*, to appear.

[4] F.Chang and L.Wu, "An Optimal Adaptive Routing Algorithm", *IEEE Trans. on Automatic Control*, pp. 690-700, Vol. AC-31, No. 8, Aug 1986.

[5] Y-C.Chow and W.H.Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System", *IEEE Trans. on Computers*, Vol. C-28, No. 5., May 1979, pp. 354-361.

[6] P.Dupuis and R.Simha, "On Sampling-controlled Stochastic Approximation", *COINS Technical Report 89-77*, Computer and Information Sciences Department, University of Massachusetts, Amherst, MA, 1989.

[7] M.C.Fu and Y.C.Ho, "Using Perturbation Analysis for Gradient Estimation, Averaging and Updating in a Stochastic Approximation Algorithm", *Proc. Winter Simulation Conf.*, 1988, pp. 509-517.

[8] R.G.Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," pp. 73-85, *IEEE Trans. on Communications*, Vol COM-25, 1977.

[9] P.W.Glynn, "Optimization of Stochastic Systems", *Proc. Winter Simulation Conference*, 1986, pp. 52-58.

[10] P.W.Glynn, "Stochastic Approximation for Monte Carlo Optimization", *Winter Simulation Conference*, 1986, pp. 356-364.

[11] B.Hajek, "Stochastic Approximation Methods for Decentralized Control of Multiaccess Communications", *IEEE Trans. on Communications*, pp. 176-184, Vol. IT-31, No.2, March 1985.

[12] G.Heal, "Planning Without Prices," *Rev. of Econ Studies*, Vol 36, pp. 346-362, 1969.

[13] P.Heidelberger and D.Towsley, "Sensitivity Analysis from Sample Paths Using Likelihoods", *COINS Technical Report TR87-131*, University of Massachusetts, Amherst, MA.

[14] Y.C.Ho, L.Servi and R.Suri, "A Class of Center-Free Resource Allocation Algorithms," *Large Scale Systems*, Vol 1, pp. 51-62, 1980.

[15] Y.C.Ho and X.R.Cao, "Perturbation Analysis and Optimization of Queueing Networks", *J. Optimization Theory & Applications*, Vol. 40, No. 4, pp. 559-582

[16] R.L.Kasyap, C.C.Blaydon and K.S.Fu, "Stochastic Approximation", in J.M.Mendel and K.S.Fu, editors, *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*, Academic Press, New York, 1970.

[17] H.Kobayashi, "Modeling and Analysis: An Introduction to System Performance and Evaluation Methodology", *Addison-Wesley*, Reading, Mass., 1978, Chapter 4.

[18] A.Kumar and F.Bonomi, "Adaptive Load Balancing in a Multiprocessor System with a Central Job Scheduler", *Proc. 2nd Int. Workshop on Applied Mathematics and Performance/Reliability Models of Computer Communication Systems*, Univ. of Rome II, pp. 173-188, 1987.

[19] J.Kurose and R.Simha, "A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems," *IEEE Trans. on Computers*, Vol. 38, No. 5, May 1989, pp. 705-717.

[20] H.J.Kushner and D.S.Clark, "Stochastic Approximation Methods for Constrained and Unconstrained Systems," *Springer Verlag*, New York, 1978.

[21] Y.T.Leung and R.Suri, "Convergence of a Single Run Simulation Optimization Algorithm", *Proc. American Control Conf.*, 1988,

[22] M.S.Meketon, "Optimization in Simulation: A Survey of Recent Results", *Proc. Winter Simulation Conf.*, 1987.

[23] P.V.McGregor and R.R.Boorstyn, "Optimal Load Sharing in a Computer Network", *Proc. of ICC*, 1975.

[24] M.I.Reiman and A.Weiss, "Sensitivity Analysis for Simulations Via Likelihood Ratios", *Operations Research*, to appear.

[25] H.Robbins and S.Munro, "A Stochastic Approximation Method," *Ann. Math. Stat.*, Vol 22, 1951, pp. 400-407

[26] R.Simha and J.F.Kurose, "Stochastic Approximation Schemes for a Load Balancing Problem", *COINS Technical Report TR 89-10*, University of Massachusetts, Amherst, MA, 1989.

[27] R.Simha and J.F.Kurose, "Stochastic Approximation Schemes for a Resource Allocation Problem with Unknown Constraints", *COINS Technical Report TR 89-11*, University of Massachusetts, Amherst, MA, 1989.

[28] R.Suri and M.A.Zazanis, "Perturbation Analysis Gives Strongly Consistent Estimates for the M/G/1 Queue", *Management Science*, Vol. 34, No. 1, Jan 1988, pp. 39-64.

[29] R.Suri, "Perturbation Analysis: The State of the Art and Research Issues Explained Via the GI/G/1 Queue", *Proc. IEEE*, Vol. 77, No. 1, Jan 1989, pp. 114-137.

[30] A.N.Tantawi and D.Towsley, "Optimal Static Load Balancing in Distributed Computer Systems", *J. ACM*, Vol. 32, No. 2, April 1985, pp. 445-465.

[31] Y.Wardi, "Simulation-based Stochastic Algorithms for Optimizing GI/G/1 Queues", manuscript, Ben Gurion University of the Negev, 1988.

[32] M.T.Wasan, "Stochastic Approximation", *Cambridge Univ. Press*, 1969.

FIGURE 1:

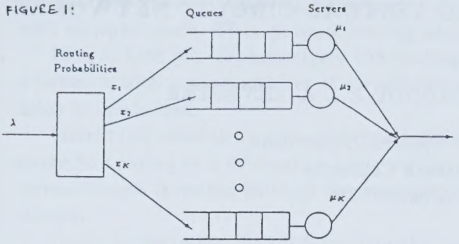

Queues
Servers
Routing
Probabilities
λ
π_1
π_2
π_K
μ_1
μ_2
μ_K

FIGURE 2: Mean delay vs. Time

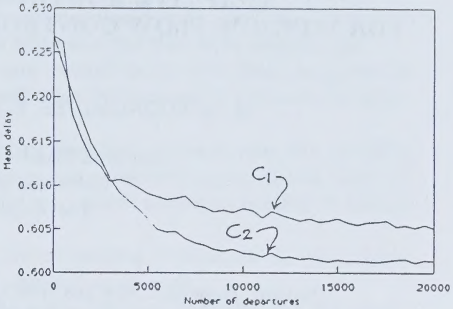


FIGURE 3: Variance of delay vs. Time

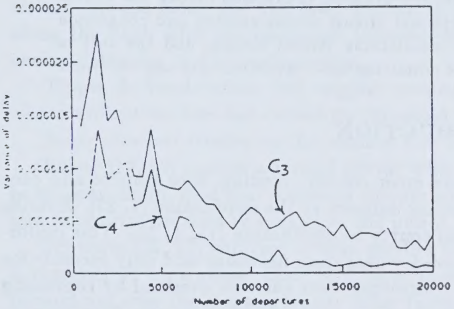


FIGURE 4: Mean delay vs. Time

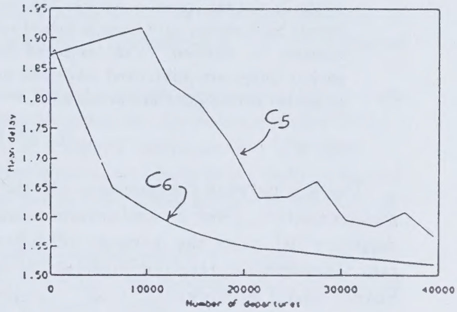


FIGURE 5: Mean delay vs. Time

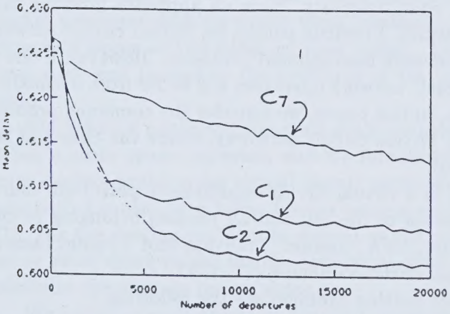
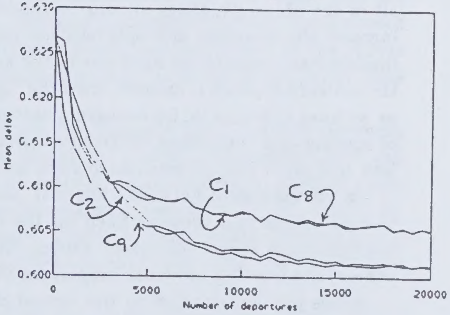


FIGURE 6: Mean delay vs. Time



ADAPTIVE ROUTING & CONGESTION CONTROL FOR WINDOW FLOW CONTROLLED VIRTUAL CIRCUIT NETWORKS

A. A. ECONOMIDES, P. A. IOANNOU & J. A. SILVESTER

Electrical Engineering - Systems Department
University of Southern California
Los Angeles, CA 90089-0781

Abstract

Routing, congestion, flow and error control are important problems for efficient network operation. Previous studies on virtual circuit network management and control consider each problem separately. However, all these problems are strongly interrelated and each affects the others. In this paper, all the above problems are simultaneously analyzed to improve virtual circuit network performance.

In contrast to previous studies, a state space nonlinear non-stationary queueing model for dynamic virtual circuit networks that considers the dynamic interaction among the virtual circuit and packet processes is introduced. Optimal virtual circuit routing and congestion controls are derived. The combined cost of maintaining virtual circuits and the cost of packet delays are minimized while the profit of admitting new virtual circuits and the profit of packet throughput are maximized.

1. INTRODUCTION

The key network management algorithms are error control, routing, flow control and congestion control. *Error control* ensures reliable packet delivery at the destination [1, 22]. *Routing* decides which route the message will follow from source to destination [15, 1, 22]. This important algorithm has received considerable attention for datagram networks and only recently for virtual circuit networks [11, 1, 22]. *Congestion control* prevents network overload by controling the traffic entering the network. *Flow control* prevents a source-destination pair from monopolizing the network and a fast sender from oversaturating a slow receiver, by controlling their traffic [15, 1, 22].

Error control, routing, congestion and flow control are strongly related problems and each one affects the others. For a more accurate network model and better network performance, all of the above problems should be modeled simultaneously. Such an approach however may increase the modeling and optimization complexity. Previous studies on virtual circuit network management consider at most one of the key network management problems. However, if one of the network controls is ignored, then the "optimal" network operation will be far from optimality, as we have shown in [4] for datagram networks. In this paper, we consider the combined problem of routing and congestion control for dynamic virtual circuit networks, where the effects of the flow and error control mechanisms are included.

In virtual circuit networks, the first packet of a virtual circuit establishes a path from source to destination that remains fixed for the duration of the call and all packets belonging to this virtual circuit follow this path. Codex, Euronet, SNA, Telenet, Transpac and Tymnet (among others) are existing networks employing virtual circuit switching [11, 1, 22].

Some previous studies on the virtual circuit routing problem are the following :

Segall [25] formulates the virtual circuit routing problem as a nonlinear programming problem on the link flows space. A virtual circuit is routed through the node with the minimum incremental delay to the destination.

---

[1] This work was supported by the National Science Foundation under Grant DMC-8452002 together with matching funds from AT & T Information Systems.

Gerla & Nilsson [12] analyze a network with a fixed number of virtual circuits, each one with window flow control. After modeling each virtual circuit as a closed chain, they approximate it with an open chain. They propose routing along the minimum first derivative length path.

Lam & Lien [16, 17] investigate the routing of a new virtual circuit with only one packet in a network with a fixed number of virtual circuits, such that the increase in the mean network delay is minimized.

Gersht [13] considers a dynamic virtual circuit routing algorithm. At each node, the available paths for routing to a destination node are the minimum hop paths to that destination node. A virtual circuit is routed through the outgoing link with the current minimum number of virtual circuits.

Gafni & Bertsekas [9, 10] formulate the virtual circuit routing problem on the path flows space. A virtual circuit is routed along the shortest marginal delay path.

Tsitsiklis & Bertsekas [29], Tsai, Tsitsiklis & Bertsekas [28] and Tsai [27] use the gradient projection method for the asynchronous virtual circuit routing problem on the path flows space and the routing decisions try to achieve some optimal target flows. They suggest that deterministic routing is better than randomized routing.

Economides, Ioannou & Silvester [3] use stochastic learning automata to adaptively route virtual circuits along the currently minimum unfinished work path, for user optimization, or along the path with the minimum increase in the number of packets or in the proportional packet delay, for system optimization.

Tipper & Sundareshan [26] suggest routing a new virtual circuit along the shortest path whose utilization does not exceed its threshold in order to minimize the average packet delay.

Some previous studies on the window flow control of a virtual circuit are the following:

Reiser [19, 20] models a virtual circuit with window flow control both as a closed chain and as an open chain, that also considers the packet delay before the packet enters the network. He also finds window sizes to maximize the power.

Schwartz [23, 22] analyzes several window flow control mechanisms for a virtual circuit that is represented by a closed chain and he proposes window sizes that maximize the ratio of the throughput over the average packet delay (power) for a virtual circuit.

In virtual circuits there are two dependent processes, the virtual circuit process and the packet process, that occur at different time scales. Previous work on virtual circuit networks do not explicitly consider the dynamic interaction between these processes. In [3], we modeled the interaction among the virtual circuit and packet processes using steady state queueing models. Now, in this paper we use a state space approach to describe both the virtual circuit and the packet processes simultaneously using nonlinear non-stationary queueing models. The network dynamics occur at two different time scales. The virtual circuit process evolves at the slower time scale and is used in the evolution of the packet process that happens at the faster time scale.

We define a nonlinear non-stationary queueing model for dynamic virtual circuit networks using a state space approach and we formulate the integrated routing, congestion, flow and error control problem for virtual circuit networks as an optimal control problem. A state space approach has also been taken for the routing problem in datagram networks [24, 18, 5, 6, 7, 14, 21]. For the routing problem in virtual circuit networks, Tipper & Sundareshan [26] use a non linear state space model for the average number of packets on each link. Their objective is to minimize the average packet delay.

We set up a multiobjective function to be optimized and we derive optimal virtual circuit routing decisions and optimal congestion controls. The combined cost of maintaining virtual circuits and the cost of packet delays are minimized while the profit of admitting new virtual circuits and the profit of packet throughput are maximized.

2. DYNAMIC VIRTUAL CIRCUIT NETWORK MODEL

We first consider a simple one hop network with multiple links. Virtual circuits arrive at a node s (according to a Poisson distribution) destined for another node d with rate $\gamma(t)$. A virtual circuit is admitted into the network with probability $\phi(t) \in [0,1]$ at time t, only if the increase in the cost that it will cause is less than the increase in the profit that it will supply to the network management, otherwise it is rejected with probability $1 - \phi(t)$. An admission controller makes these decisions and updates this admission probability $\phi(t)$ according to the current network state. Then the actual virtual circuit arrival rate into the network is $\gamma(t) * \phi(t)$.

If the virtual circuit is accepted, then it is routed to node d through one of the L links that connect these two nodes with probability $P_i(t) \in [0,1]$ through link i, $i = 1, ..., L$, $\sum_i P_i(t) = 1$ at time t. A router makes these decisions and updates these routing probabilities according to the current network state. Then the virtual circuit arrival rate to link i is $\gamma(t) * \phi(t) * P_i(t)$.

Finally, each virtual circuit stays in the network for some time duration generally distributed with mean $1/\delta(t)$ and then terminates. So, we can model every link i for the virtual circuit process as an $M/G/\infty$ queue with arrival rate $\gamma(t) * \phi(t) * P_i(t)$ and mean service time $1/\delta(t)$ (Fig. 1). Subsequently, we will introduce a state space approach to model the dynamic evolution of the virtual circuit processes.

The expected number of virtual circuits on link i at time t, $V_i(t) \geq 0$, $i = 1, ...L$ increases during Δt by the expected number of virtual circuits that arrive during this period, $\gamma(t) * \phi(t) * P_i(t) * \Delta t$, minus the expected number of virtual circuits that depart during this period, $\delta(t) * V_i(t) * \Delta t$. So, the virtual circuit process at link i is described by

$$\begin{aligned} V_i(t+\Delta t) &= V_i(t) + \gamma(t) * \phi(t) * P_i(t) * \Delta t - \delta(t) * V_i(t) * \Delta t \qquad i = 1, ..., L \ \ or \\ \dot{V}_i(t) &= \gamma(t) * \phi(t) * P_i(t) - \delta(t) * V_i(t) \quad for \ \ \Delta t \to 0, \qquad i = 1, ..., L \end{aligned}$$

Next we describe the evolution of the packet process into the network. Let $r(t)$ be the packet arrival rate per virtual circuit at time t (Poisson distribution). If there are $V_i(t)$ virtual circuits on link i at time t, then the total packet arrival rate to link i is $r(t) * V_i(t)$, since all packets belonging to a virtual circuit are transmitted through the same link.

Upon arriving to link i, a packet is not immediately accepted for transmission, but first stays in the input queue for link i (Fig. 2), where the expected number of packets at time t is $I_i(t) \geq 0$, $i = 1, ..., L$. A window flow control mechanism permits at most $W_i(t)$ packets to exist on link i (either in the transmission queue or being actually transmitted) at time t, where $W_i(t) \geq 0$, $i = 1, ..., L$. A packet that is to be transmitted through link i waits in the input queue for this link until a permit comes back from a successfully received packet at node d. Let $A_i(t)$ be the expected number of permits available at link i at time t, $A_i(t) \geq 0$, $i = 1, ..., L$.

Let the expected number of packets for link i, $N_i(t) \geq 0$, $i = 1, ..., L$, be equal to the expected number of packets in the input queue for link i, $I_i(t)$, plus the expected number of packets in transit at link i, i.e. the window size $W_i(t)$, minus the expected number of available permits $A_i(t)$

$$N_i(t) = I_i(t) + W_i(t) - A_i(t) \qquad i = 1, ..., L$$

When a permit becomes available, the packet is accepted into the transmission queue for link i and waits for its transmission (on first-come first-served scheduling). When reaching the head of this queue, the packet is transmitted. Let τ_i be the mean packet transmission time at link i (general distribution) and η_i be the packet propagation delay at link i (general distribution).

However, with probability $e_i(t)$ for link i at time t, $e_i(t) \in [0,1]$, the packet may arrive in error (or perhaps never arrive) at node d. If the packet arrives in error, node s is notified with a negative acknowledgment (NACK) by node d and let θ_i be the NACK/time-out delay for link i (general distribution). Then node s retransmits it. If it never arrives, eventually node s times out after time θ_i and retransmits it.

If a packet successfully arrives at node d, a positive acknowledgment (ACK) is sent back to node s where it arrives after time α_i (general distribution). This ACK also acts as a permit for the transmission of another packet through this link. However, the ACK may also fail, with probability $e_i^{'}(t) \in [0,1]$, and therefore node s will retransmit the packet after time β_i (general distribution).

Next, we find the mean time $x_i(t)$ that a packet spends on link i after starting transmission (no queueing delays) at time t (Fig. 2 & 3). This will be the sum of the mean transmission delay τ_i, plus the propagation delay η_i, plus, with error probability $e_i(t)$, the mean time spent when there is a packet error, plus, with no error probability $1 - e_i(t)$, the mean time spent for the ACK. The mean time spent when there is a packet error is the sum of the mean NACK/time-out delay plus the mean retransmission time $x_i(t)$. The mean time spent for the ACK is the sum of the mean ACK delay α_i plus the mean NACK/time-out delay $\beta_i(t)$ and mean retransmission time $x_i(t)$ in case the ACK fails with probability $e_i^{'}(t)$

$$x_i(t) = \tau_i + \eta_i + e_i(t) * (\theta_i + x_i(t)) + (1 - e_i(t)) * [\alpha_i + e_i^{'}(t) * (\beta_i(t) + x_i(t))]$$

Next, we assume that the packet service times on link i are independent exponentially distributed random variables with rate $mu_i(t) = 1/x_i(t)$. Thus:

$$\mu_i(t) = \frac{1 - e_i(t) - [1 - e_i(t) * e_i^{'}(t)]}{\tau_i + \eta_i + e_i(t) * \theta_i + [1 - e_i(t)] * [\alpha_i + e_i^{'}(t) * \beta_i]}$$

In general, it is not true that the packet service times will be independent exponentially distributed, but we make this assumption, in order to have a simple model that captures the effect of the link error rate and ACK process on the packet service time.

Then the Markov chain with state $N_i = I_i + W_i - A_i$ is an $M/M/1$ queue (Fig. 3 & 4). So, for the packet process, we model each link i as an $M/M/1$ queue, with packet arrival rate $r(t) * V_i(t)$ and mean service time $1/\mu_i(t)$.

Let $\rho_i(N_i(t))$ be an approximation of the instantaneous utilization for link i such that the packet departure rate from link i at time t is $\mu_i(t) * \rho_i(N_i(t))$.

Then the expected number of packets at link i at time t, $N_i(t)$, will increase during Δt by the expected number of packets that will arrive during this period, $r(t) * V_i(t) * \Delta t$, minus the expected number of packets that will depart during this period, $\mu_i(t) * \rho_i(t)$. Since, the link utilization $\rho_i(N_i(t))$, is a nonlinear function of the number of packets at link i, $N_i(t)$, the packet process at link i is described by a nonlinear dynamic model

$$\begin{aligned} N_i(t + \Delta t) &= N_i(t) + r(t) * V_i(t) * \Delta t - \mu_i(t) * \rho_i(N_i(t)) * \Delta t \quad i = 1, ..., L \quad or \\ \dot{N}_i(t) &= r(t) * V_i(t) - \mu_i(t) * \rho_i(N_i(t)) \quad for \;\; \Delta t \to 0, \quad i = 1, ..., L \end{aligned}$$

A similar nonlinear non-stationary model for the average number of packets at a link has also been considered by Filipiak [5, 6, 7, 8] and by Tipper & Sundareshan [26]. However, in our model we make the packet arrival rate depend on the expected number of virtual circuits, i.e. the network state. In this way, we introduce the dynamic interaction among the virtual circuit and packet processes into our model. Although a nonlinear model will introduce extra complexity in the optimization procedures, we prefer to have a nonlinear model than a simpler linear model, since the queueing processes are extremely nonlinear processes.

Having described the dynamics of the virtual circuit and the packet processes, we define the network state to be

$$\mathbf{X}(t) = [V_1(t), ..., V_L(t), N_1(t), ..., N_L(t)]^T$$

In the next section, we will use this nonlinear dynamic model for non-stationary virtual circuit networks to formulate and solve the combined routing and congestion control for window flow controlled virtual circuit networks as an optimal control problem. Note, also that the effect of the variable link error rates has been included into the model.

3. MULTI - OBJECTIVE FUNCTION

In this section, we formulate the integrated routing, congestion, window flow and error control problem for virtual circuit networks as an optimal control problem. An optimal control approach has also been taken by Filipiak [5, 8], for the routing and congestion control problems, for datagram networks. He minimized the average packet delay and maximized the gain due to acceptance of packets into the network. Also, Tipper & Sundareshan [26] formulated the routing problem for virtual circuit networks as an optimal control problem. Their objective function was also to minimize the average packet delay.

Here, we set up a multiobjective function to be optimized, for the integrated problem. We would like to minimize the cost of setting up and maintaining the virtual circuits inside the network, while also maximizing the profit of admitting virtual circuits into the network. We would also like to minimize the cost of having the packets inside the network, while also maximizing the profit of servicing packets. To accomplish this, we define the following costs and profits :

$C_{V,i}(t)$: cost per virtual circuit for link i at time t, (for example, the cost of setting up and maintaining the virtual circuit path through link i).
$C_{\phi}(t)$: profit of admitting a new virtual circuit into the network at time t, or cost of rejecting a new virtual circuit at time t.
$C_{N,i}(t)$: cost per packet for link i at time t.
$C_{\mu,i}(t)$: profit of servicing a packet at link i at time t.

Then our problem is to

minimize $$\int_0^T \{ \sum_i C_{V,i}(t) * V_i(t) - C_{\phi}(t) * \phi(t) * \gamma(t) + + \sum_i C_{N,i}(t) * N_i(t) - \sum_i C_{\mu,i}(t) * \mu_i(t) * \rho_i(t) \} dt$$

with respect to the routing probabilities $P_i(t) \quad i = 1, ..., L,$
the congestion control parameter $\phi(t)$

such that

$$\dot{\mathbf{X}}(t) = \begin{bmatrix} \gamma(t) * \phi(t) * P_1(t) - \delta(t) * V_1(t) \\ \gamma(t) * \phi(t) * P_2(t) - \delta(t) * V_2(t) \\ \cdots \\ \gamma(t) * \phi(t) * P_L(t) - \delta(t) * V_L(t) \\ r(t) * V_1(t) - \mu_1(t) * \rho_1(t) \\ r(t) * V_2(t) - \mu_2(t) * \rho_2(t) \\ \cdots \\ r(t) * V_L(t) - \mu_L(t) * \rho_L(t) \end{bmatrix} \qquad \sum_i P_i(t) = 1, \quad P_i(t) \geq 0, \quad \forall i \qquad \phi(t) \in [0,1]$$

The first term of the objective function is the sum of the average cost of setting up and maintaining $V_i(t)$ virtual circuits on every link i, $i = 1, ..., L$, at time t. The second term of the objective function is the average loss of not admitting new virtual circuits into the network at time t. The third term of the objective function is the sum of the average cost of having $N_i(t)$ packets on every link i, $i = 1, ..., L$, at time t. When this cost is integrated over a time period, it reflects the average cost of packet delay during that time period. Finally, the last term of the objective function is the negative profit of packet throughput $\mu_i(t) * \rho_i(t)$ on every link i, $i = 1, ..., L$, at time t. To proceed, we assume that $\rho_i(t)$ is defined for $N_i(t) \geq 0$, is concave, monotonically increasing and twice differentiable in N_i with $\lim_{N_i \to \infty} \rho_i(N_i) = 1$.

Necessary conditions for optimality are provided by the Pontryagin maximum principle [30, 2]. The Hamiltonian function of the state X, the controls P, ϕ and the costate variables Q at time t is

$$
\begin{aligned}
H(X,P,\phi,Q,t) \;=\; & \sum_i C_{V,i}(t) * V_i(t) - C_\phi(t) * \phi(t) * \gamma(t) \;+ \\
+ & \sum_i C_{N,i}(t) * N_i(t) - \sum_i C_{\mu,i}(t) * \mu_i(t) * \rho_i(t) \;+ \\
+ & \sum_i Q_{V,i}(t) * [\gamma(t) * \phi(t) * P_i(t) - \delta(t) * V_i(t)] \;+ \\
+ & \sum_i Q_{N,i}(t) * [r(t) * V_i(t) - \mu_i(t) * \rho_i(t)]
\end{aligned}
$$

$$
\textit{such that } \sum_i P_i(t) = 1, \quad P_i(t) \geq 0, \quad \forall i \quad \phi(t) \in [0,1].
$$

also, the costate variables must satisfy the adjoint equations

$$
\dot{Q}_{V,i}(t) = -\frac{\partial H(X,P,\phi,Q,t)}{\partial V_i(t)} = -\{\, C_{V,i}(t) - Q_{V,i}(t) * \delta(t) + Q_{N,i}(t) * r(t) \,\}
$$

$$
\begin{aligned}
\dot{Q}_{N,i}(t) = -\frac{\partial H(X,P,\phi,Q,t)}{\partial N_i(t)} = & -\{\, C_{N,i}(t) - C_{\mu,i}(t) * \mu_i(t) * \frac{d\rho_i(t)}{dN_i(t)} - \\
& -Q_{N,i}(t) * \mu_i(t) * \frac{d\rho_i(t)}{dN_i(t)} \,\}
\end{aligned}
$$

In this section, we formulated the integrated virtual circuit routing, congestion control, window flow control and the link error rate effect on the performance as an optimal control problem. In the next section we will find the optimum feedback control policies.

4. ADAPTIVE OPTIMAL CONTROL POLICIES

In this section we find the optimum virtual circuit routing and congestion control policies. The minimization of the Hamiltonian with respect to the routing probabilities is equivalent to the following minimization problem

$$
\begin{array}{ll}
\textit{minimize} & \sum_i Q_{V,i}(t) * P_i(t) \\
\textit{with respect to} & P_i(t) \quad \forall i \\
\textit{such that} & \sum_i P_i(t) = 1, \quad P_i(t) \geq 0 \quad \forall i
\end{array}
$$

A similar problem has also been considered by Filipiak [5, 6, 8], for routing packets in datagram networks and by Tipper & Sundareshan [26] for virtual circuit routing. However, the state of our system depends both on the expected number of virtual circuits and packets in the network and we also optimize a multiobjective function. Therefore, our costate variables are different than theirs. So, let the minimum costate variable be $Q^*_V(t) = \min_i\{Q_{V,i}(t)\}$, then the optimum routing probabilities will be:

$$
P^*_i(t) \begin{cases} > 0 & \text{only if } Q_{V,i}(t) = Q^*_V(t). \\ = 0 & \text{o.w.} \end{cases}
$$

Therefore link i will be used only if it has the minimum costate variable $Q_{V,i}(t) = Q^*_V(t)$. If only one link i achieves the minimum costate variable $Q^*_V(t)$, then $P^*_i(t) = 1$. However, if more than one link achieves the minimum costate variable $Q^*_V(t)$, then for these links $0 < P^*_i(t) < 1$ and for the rest links $P^*_j(t) = 0 \quad \forall j \neq i$.

Next, we solve the problem for the steady state ($T \to \infty$). The mean number of packets for link i at steady state is $N_i = \rho_i/(1-\rho_i)$. Rewriting, we have the utilization of link i at steady state $\rho_i = N_i/(1+N_i)$. At steady state, the costate variables must satisfy

$$\dot{Q}_{N,i} = 0 \Rightarrow \quad -\{C_{N,i} - C_{\mu,i} * \mu_i * \frac{d\rho_i}{dN_i} - Q_{N,i} * \mu_i * \frac{d\rho_i}{dN_i}\} = 0$$

$$Q_{N,i} = \frac{C_{N,i}}{\mu_i * \frac{d\rho_i}{dN_i}} - C_{\mu,i} = \frac{C_{N,i} * (1+N_i)^2}{\mu_i} - C_{\mu,i}$$

$$\dot{Q}_{V,i} = 0 \Rightarrow \quad -\{C_{V,i} - Q_{V,i} * \delta + Q_{N,i} * r\} = 0$$

$$Q_{V,i} = \frac{C_{V,i} + Q_{N,i} * r}{\delta} = \frac{C_{V,i} + C_{N,i} * (1+N_i)^2/\mu_i - C_{\mu,i}}{\delta}$$

Routing Rule : a new virtual circuit is routed through the link that has the minimum costate variable:

$$\frac{C_{V,i} + C_{N,i} * (1+N_i)^2/\mu_i - C_{\mu,i}}{\delta} = \min_j\{\frac{C_{V,j} + C_{N,j} * (1+N_j)^2/\mu_j - C_{\mu,j}}{\delta}\}$$

A simple example for routing to two links with $C_{V,1} = C_{V,2} = 0$, $C_{\mu,1} = C_{\mu,2} = 0$ and $\mu_2 = 4 * \mu_1$ is given in Fig. 5.

Now, we substitute these optimum routing probabilities P_i^* $\forall i$ into the Hamiltonian and minimize the Hamiltonian with respect to the admission probability ϕ

minimize $\quad [-C_\phi + \sum_i Q_{V,i} * P_i^*] * \phi$
with respect to $\quad \phi$
such that $\quad \phi \in [0,1]$.

When the routing probabilities achieve their optimum values P_i^* $\forall i$, we have

$$\sum_i Q_{V,i} * P_i^* = Q_V^*$$

Then the minimization of the Hamiltonian with respect to the admission probability ϕ is equivalent to

minimize $\quad [-C_\phi + Q_V^*] * \phi$
with respect to $\quad \phi$
under the constraints $\quad \phi \in [0,1]$.

The optimum admission probability is

$$\phi^* = \begin{cases} 0 & \text{for } Q_V^* > C_\phi \\ 1 & \text{for } Q_V^* < C_\phi \\ 0 \le \phi \le 1 & \text{for } Q_V^* = C_\phi \end{cases}$$

Admission Rule : a new virtual circuit is admitted into the network only if

$$\min_i\{\frac{C_{V,i} + C_{N,i} * (1+N_i)^2/\mu_i - C_{\mu,i}}{\delta}\} < C_\phi$$

In this section, we have derived the optimum virtual circuit routing and congestion control policies that also depend on the window flow control parameters and the link error rates.

5. CONCLUSIONS

In this paper, we have presented *nonlinear non-stationary queueing models of virtual circuit networks*, by explicitly considering the interaction among the virtual circuit and packet processes. We *formulated the integrated virtual circuit routing, congestion, flow and error control* problem as an optimal control problem. We set up a *multiple objective function* and we solved it using the Pontryagin maximum principle. Finally, we derived *feedback optimal control policies* for virtual circuit network management.

We are currently working on simulating the proposed nonlinear non-stationary queueing models and optimal control policies, as well as extending them for virtual circuit networks with end to end window flow control. We are also studying the impact of the window flow control parameters and investigating under what conditions the assumption of independent exponential service times on the links is valid. We will use this queueing theory-based state space approach for the integrated virtual circuit network management and control.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Vikram Saksena of AT & T Bell Laboratories for suggesting the virtual circuit routing problem and for many useful discussions.

References

[1] D.P. Bertsekas and R. Gallager. *Data Networks.* Prentice Hall, 1987.

[2] A.E. Jr. Bryson and Y. C. Ho. *Applied optimal control.* Hemisphere Publ. Co, 1975.

[3] A.A. Economides, P.A. Ioannou, and J.A. Silvester. Decentralized adaptive routing for virtual circuit networks using stochastic learning automata. *Proc. of IEEE Infocom 88 Conference*, pp. 613-622, IEEE 1988.

[4] A.A. Economides and J.A. Silvester. Optimal routing in a network with unreliable links. *Proc. of IEEE Computer Networking Symposium*, pp. 288-297, IEEE 1988.

[5] J. Filipiak. Optimal control of store-and-forward networks. *Optimal Control Applications & Methods, Vol. 3, pp. 155-176*, 1982.

[6] J. Filipiak. Dynamic routing in queueing systems with a multiple service facility. *Operations Research, Vol. 32, No. 5, pp. 1163-1180*, 1984.

[7] J. Filipiak. Analysis and synthesis of routing and flow control in a nonstationary environment. *Proc. of International Conference on Computer Communications, pp. 446-451*, 1985.

[8] J. Filipiak. *Modelling and Control of Dynamic Flows in Communication Networks.* Springer-Verlag, 1988.

[9] E.M. Gafni. and D.P. Bertsekas. Path assignment for virtual circuit routing. *Proc. ACM SIGCOMM 1983 Conference, pp. 21-25*, ACM 1983.

[10] E.M. Gafni and D.P. Bertsekas. Asymptotic optimality of shortest path routing algorithms. *IEEE Trans. on Information Theory, Vol. IT-33, No. 1, pp. 83-90*, Jan. 1987.

[11] M. Gerla. Packet, circuit and virtual circuit switching. *in W. Chou (ed.) "Computer Communications vol II : Systems and Applications", pp. 222-267*, Prentice Hall 1985.

[12] M. Gerla and P.O. Nilsson. Routing and flow control interplay in computer networks. *Proc. International Conference on Computer Communications, pp. 84-89*, 1980.

[13] A. Gersht. Analytical model of dynamic routing in virtual circuit packet switching network. *Proc. International Conference on Computer Communications, pp.76-80*, IEEE 1982.

[14] B. Hajek and R.G. Ogier. Optimal dynamic routing in communication networks with continuous traffic. *Networks, Vol. 14, pp. 457-487*, 1984.

[15] L. Kleinrock. *Queueing Systems, Vol. 2: Applications.* J. Wiley & Sons, 1976.

[16] S.S Lam and Y.C.L. Lien. Modeling and analysis of flow controlled packet switching networks. *Proc. 7th Data Communications Symposium, pp.98-107*, IEEE 1981.

[17] S.S. Lam and Y.L. Lien. Optimal routing in networks with flow-controlled virtual channels. *Proc. Computer Networks Performance Evaluation, pp.38-46*, ACM 1982.

[18] F.H. Moss and A. Segall. An optimal control approach to dynamic routing in networks. *IEEE Tr. on Automatic Control, Vol.AC -27, No. 2, pp. 329-339*, Apr. 1982.

[19] M. Reiser. A queueing network analysis of computer communication networks with window flow control. *IEEE Tr. on Communications, Vol. COM-27, No. 8, pp. 1199-1209*, Aug. 1979.

[20] M. Reiser. Admission delays on virtual routes with window flow control. *G. Pujolle (ed.) Performance of Data Communication Systems and their Applications, pp. 67-76*, North Holland Publ. Co. 1981.

[21] G. Sasaki and B. Hajek. Optimal dynamic routing in single commodity networks by iterative methods. *IEEE Tr. on Communications, Vol. COM-35, No. 11, pp. 1199-1205*, Nov. 1987.

[22] M. Schwartz. *Telecommunication Networks : Protocols, Modeling and Analysis.* Addison Wesley, 1988.

[23] M. Schwartz. Performance analysis of the sna virtual route pacing control. *IEEE Tr. on Communications, Vol. COM-30, No. 1, pp. 172-184*, Jan. 1982.

[24] A. Segall. The modeling of adaptive routing in data communication networks. *IEEE Tr. on Communications, Vol. COM-25, No. 1, pp. 85-95*, Jan. 1977.

[25] A. Segall. Optimal distributed routing for virtual line-switched networks. *IEEE Tr. on Communications, Vol.COM-27, No. 1, pp.*, Jan. 1979.

[26] D. Tipper and M.K. Sundareshan. Optimal threshold routing of virtual circuits in data networks. *ORSA/TIMS Joint National Meeting*, 1988.

[27] W.K. Tsai. Convergence of gradient projection routing methods in an asynchronous stochastic quasi-static virtual circuit network. *IEEE Tr. on Automatic Control, Vol. AC-34, No. 1, pp. 20-33*, Jan. 1989.

[28] W.K. Tsai, J.N. Tsitsiklis, and D.P. Bertsekas. Some issues in distributed asynchronous routing in virtual circuit data networks. *Proc. IEEE 25th Conf. on Decision and Control, pp.1335-1337*, IEEE Dec. 1986.

[29] J.N. Tsitsiklis and D.P. Bertsekas. Distributed asynchronous optimal routing in data networks. *IEEE Tr. on Automatic Control, Vol. AC-31, No. 4, pp. 325-332*, Apr. 1986.

[30] P.P. Varaiya. *Notes on optimization.* Van Nostrand Reinhold Co., 1972.

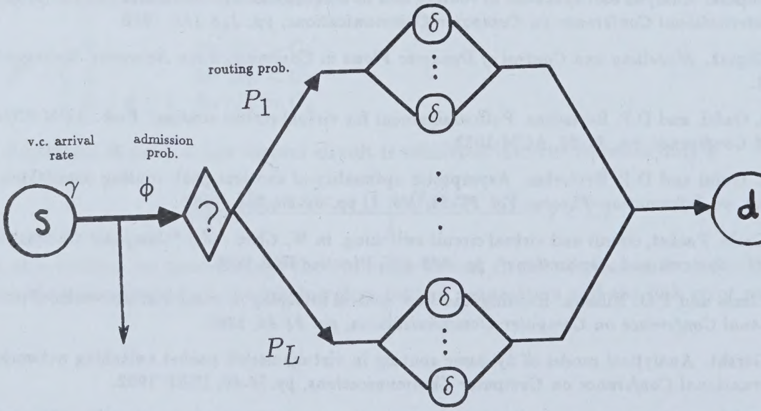


Fig. 1 Virtual circuit process.

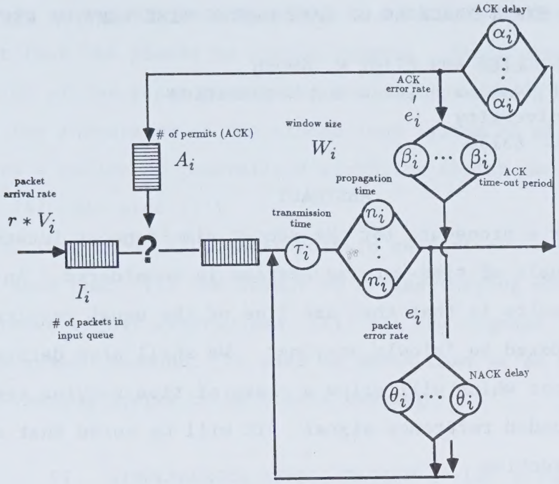

ACK delay
α_i
α_i
ACK error rate
e_i
# of permits (ACK)
A_i
window size
W_i
β_i
β_i
ACK time-out period
packet arrival rate
$r * V_i$
propagation time
transmission time
n_i
τ_i
n_i
?
I_i
# of packets in input queue
e_i
packet error rate
NACK delay
θ_i
θ_i

Fig. 2 Packet process for link *i*.

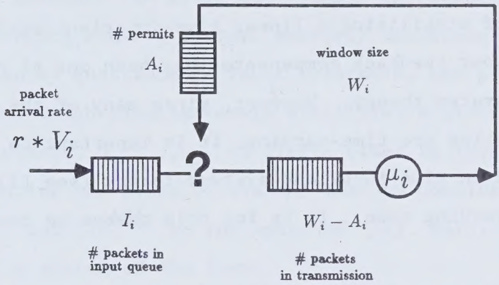

# permits
A_i
window size
W_i
packet arrival rate
$r * V_i$
?
μ_i
I_i
$W_i - A_i$
# packets in input queue
# packets in transmission

Fig. 3 Approximate model for link *i*.

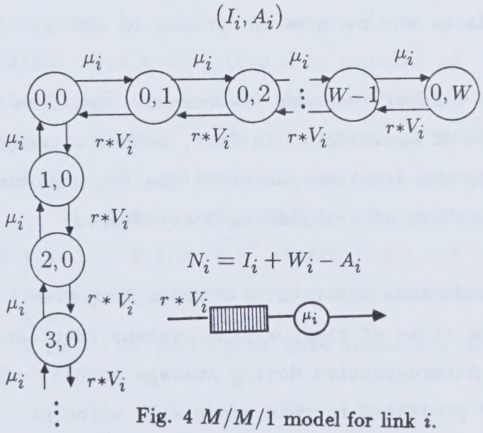

(I_i, A_i)
μ_i
μ_i
μ_i
μ_i
0,0
0,1
0,2
W−1
0,W
$r*V_i$
$r*V_i$
$r*V_i$
$r*V_i$
μ_i
1,0
μ_i
$r*V_i$
2,0
μ_i
$r*V_i$
3,0
μ_i
$r*V_i$
$N_i = I_i + W_i - A_i$
$r * V_i$
μ_i

Fig. 4 $M/M/1$ model for link *i*.

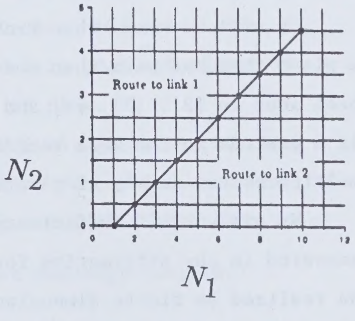


Fig. 5 Threshold routing to two links.

STABILIZATION AND EXACT TRACKING OF FAST MOVING TIME-VARYING SYSTEMS

PAUL R. BOUTHELLIER AND BIJOY K. GHOSH
Department of Systems Science and Mathematics
Washington University
St. Louis, MO 63130

ABSTRACT

In this paper a procedure for the robust simultaneous internal stabilization of a r-tuple of time-varying systems is considered. An important feature of our results is that they are free of the usual requirement that the systems considered be "slowly varying". We shall also derive a simple feedback compensator which will allow a class of time-varying systems to stably track a bounded reference signal. It will be noted that all our results are constructive.

I. INTRODUCTION

The problem of stabilizing a linear time-invariant system by a linear time-invariant output feedback compensator has been one of the most important problems in system theory. However, since many of the systems considered in practice are time-varying, it is important to generalize results well known in time-invariant systems theory (see [2], [3], [4], [5]) to the time-varying case. It is for this reason we consider the following problem:

Problem 1: (Simultaneous Stabilizability Problem) Given a r-tuple of discrete-time multi-input multi-output linear time-varying dynamical systems, does there exists a linear time-varying dynamic compensator which simultaneously (internally) stabilizes the r-tuple of systems in the closed-loop?

It will be noted that Problem 1 is of interest in order to stabilize a plant that has more than one mode of operation. In fact, as has already been seen in [2], [3], [4] and [5], the simultaneous stabilization problem is a prelude to the more general problem of stabilizing a structured multiparameter family of plants.

We will obtain sufficiency conditions under which Problem 1 can be answered in the affirmative for the class of time-varying systems that can be realized as finite dimensional Autoregressive Moving Average (ARMA) models. We remark that the method presented in this paper will allow us to explicitly construct the compensator and is free of the ubiquitous requirement that the plants be slowly varying. Since obtaining the precise value of the plant parameters for a time-varying system is often difficult, the robustness of the closed-loop system is ascertained by appealing to a notion of generalized stability region described by the authors in [6] (see also [7]).

We will then show that, under certain conditions, a compensator can be derived such that (i) the output of a time-varying plant tracks a given reference signal exactly and (ii) all the signals in the closed-loop system remain bounded. It will be noted that we do not require the given time-varying system to be slowly varying.

II. SIMULTANEOUS STABILIZATION OF LTV SYSTEMS

This section concentrates of the basic problem of understanding the extent to which feedback can alter the dynamic characteristics of a linear discrete time-varying control system, possibly operating in one or more modes. Motivated by questions in fault tolerance, the problem of designing a compensator which simultaneously stabilizes a given r-tuple of m input p output plants $G_1,\dots,G_r$ of finite order is investigated.

We first define the class of systems that we consider.

Definition 2.1: Let $\mathbb{R}^{i\times j}$ be the space of $i\times j$ matrices whose elements belong to $\mathbb{R}$. A system of the form

$$y(k) + \sum_{i=1}^{\ell} D(i,k)y(k-i) = \sum_{i=1}^{\ell} N(i,k)u(k-i) \tag{2.1}$$

where $D(i,k)\in\mathbb{R}^{p\times p}$ and $N(i,k)\in\mathbb{R}^{p\times m}$ $i=1,2,\dots,\ell$ $\forall$ $k\in Z$, the set of integers, is defined to be a m input p output ARMA model of lag ℓ, if either $D(\ell,k)\neq[0]$ (the zero matrix) or $N(\ell,k)\neq[0]$ for some k. We shall also assume that (A1) $y(\cdot)$ and $u(\cdot)$ have zero initial conditions, i.e. $y(k)=\underline{0}\in\mathbb{R}^p$ and $u(k)=\underline{0}\in\mathbb{R}^m$ $\forall$ $k<0$ and (A2) $\|D(i,\cdot)\|$, $\|N(i,\cdot)\|$ $i=1,2,\dots,\ell$ are uniformly bounded functions of time.

It will be noted that by considering $(D(1,k),D(2,k),\dots,D(\ell,k);$ $N(1,k),\dots,N(\ell,k))\in\mathbb{R}^N$ $(N=\ell(p^2+pm))$ as a column vector for each time k, time-varying systems of the form (2.1) are elements of the product space $\displaystyle\mathop{X}_{j=-\infty}^{\infty}\mathbb{R}^N$. We shall give this space the product topology (see [8]).

The simultaneous stabilization problem can now be described as follows:

Problem 2.2: Given a r-tuple of $p\times m$ systems

$$y(k) + \sum_{i=1}^{\ell} D_j(i,k)y(k-i) = \sum_{i=1}^{\ell} N_j(i,k)u(k-i) \qquad (2.2)$$

denoted by $\{G_j(k)\}_{j=1}^{r}$, each of lag ℓ satisfying (A1) and (A2), find a $m\times p$ compensator $C(k)$ of minimal lag q defined by

$$u^*(k) + \sum_{i=1}^{q} D(i,k)u^*(k-i) = \sum_{i=0}^{q} N(i,k)y(k-i) \qquad (2.3)$$

such that the closed-loop systems defined by Figure 1 below, where we assume $u_2(k)=\underline{0}\in \mathrm{IR}^p$ and $y_2(k)=\underline{0}\in \mathrm{IR}^m$ V $k<0$, are simultaneously internally stable (i.e., all the signals in each closed-loop system are bounded).

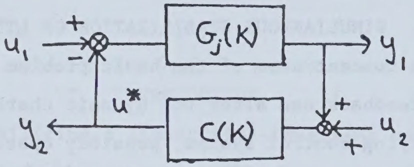


Figure 1

To solve Problem 2.2 we define two operators which will prove useful. Some of the properties of the operators will then be given. Where $a(k)\in \mathrm{IR}^{p\times m}$ V $k\in Z$ define [9] $z^{-\ell}\cdot$ by $z^{-\ell}\cdot a(k)=a(k-\ell)$ and $z^{-\ell}o$ by $z^{-\ell}oa(k)=a(k-\ell)z^{-\ell}$. The following proposition, which lists the key properties of the above operators, may be verified by direct computation and hence the proof is omitted.

Proposition 2.3 [1]: Where $\mathrm{IR}^{p\times m}[z^{-1}]$ is defined to be the set of power series in z^{-1} with coefficients taking values in $\mathrm{IR}^{p\times m}$:

i.) If $D(k,z^{-1})=I+\sum_{i=1}^{\ell} D(i,k)z^{-i}(\in \mathrm{IR}^{p\times p}[z^{-1}])$ there exists a unique inverse $D^{-1}(k,z^{-1})\in \mathrm{IR}^{p\times p}[z^{-1}]$ such that $D^{-1}(k,z^{-1})oD(k,z^{-1}) = D(k,z^{-1})oD^{-1}(k,z^{-1}) = I$ V $k\in Z$.

ii.) Let $A_i(k,z^{-1})\in \mathrm{IR}^{p_i\times p_{i+1}}$ $i=1,2,3$ and $y(k)\in \mathrm{IR}^{p_4}$ V $k\in Z$ where p_i $i=1,2,3,4$ are finite positive integers, then $A_1(k,z^{-1})\cdot[A_2(k,z^{-1})\cdot(A_3(k,z^{-1})\cdot y(k))]=[A_1(k,z^{-1})o(A_2(k,z^{-1})oA_3(k,z^{-1}))]\cdot y(k)$

iii.) o is associative.

Using the operators defined above we can write the r-tuple of plants $\{G_j\}_{j=1}^{r}$ given by (2.2) as

$$(I + \sum_{i=1}^{\ell} D_j(i,k)z^{-i})\cdot y(k) = (\sum_{i=1}^{\ell} N_j(i,k)z^{-i})\cdot u(k) \tag{2.4}$$

$j=1,2,\ldots,r$ or in transfer function form [9] as

$$y(k) = [D_j^{-1}(k,z^{-1}) \circ N_j(k,z^{-1})]\cdot u(k) \tag{2.5}$$

where

$$D_j(k,z^{-1}) = I+ \sum_{i=1}^{\ell} D_j(i,k)z^{-i};\ N_j(k,z^{-1})= \sum_{i=1}^{\ell} N_j(i,k)z^{-i} \tag{2.6}$$

$j=1,\ldots,r$. Similarily, the compensator $C(k)$ given by (2.3) may be written as

$$u^*(k) = [D^{-1}(k,z^{-1}) \circ N(k,z^{-1})]\cdot y(k) \tag{2.7}$$

where

$$D(k,z^{-1}) = I + \sum_{i=1}^{q} D(i,k)z^{-i};\ N(k,z^{-1}) = \sum_{i=0}^{q} N(i,k)z^{-i} \tag{2.8}$$

For the sake of computation we shall write $C(k,z^{-1})$ in the form

$$C(k,z^{-1}) = \bar{N}(k,z^{-1}) \circ \bar{D}^{-1}(k,z^{-1}) \tag{2.9}$$

where

$$\bar{N}(k,z^{-1}) = \sum_{i=0}^{\bar{q}} \bar{N}(i,k)z^{-i};\ \bar{D}(k,z^{-1}) = I+ \sum_{i=1}^{\bar{q}} \bar{D}(i,k)z^{-i} \tag{2.10}$$

To show this alternate representation of the compensator is feasible we state the following proposition.

Proposition 2.4 [1]: Given a generic m×p plant of lag $\bar{q}$ as defined by (2.9)-(2.10) there exists operator polynomials $N(k,z^{-1})$ and $D(k,z^{-1})$ of lag q such that $[D^{-1}(k,z^{-1}) \circ N(k,z^{-1})]\cdot y(k)=[\bar{N}(k,z^{-1}) \circ \bar{D}^{-1}(k,z^{-1})]\cdot y(k)$ for all sequences $y(\cdot)\in \mathbb{R}^p$ satisfying $y(k)=\underline{0}\ \forall\ k<0$ where $qm \geq \bar{q}p$.

Note: By generic we shall mean that the set of plants for which Proposition 2.4 holds can be written as the intersection of a countable number of open and dense sets in $\mathop{X}_{j=-\infty}^{\infty} \mathbb{R}^N (N=\bar{q}(p^2+mp)+mp)$. It should also be noted that the m×p system $C(k,z^{-1})$ given by (2.9)-(2.10) always admits a state-space realization [10].

With the r-tuple of plants (2.2) written as (2.5)-(2.6) and the compensator (2.3) written as (2.9)-(2.10) it is shown in [1] that the closed-loop systems defined by Figure 1 can be expressed (omitting the z^{-1} and k terms) as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \bar{D}o(D_j o\bar{D}+N_j o\bar{N})^{-1} oN_j & -I+\bar{D}o(D_j o\bar{D}+N_j o\bar{N})^{-1} oD_j \\ \bar{N}o(D_j o\bar{D}+N_j o\bar{N})^{-1} oN_j & \bar{N}o(D_j o\bar{D}+N_j o\bar{N})^{-1} oD_j \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.11}$$

From (2.11) it is clear that sufficient conditions for the simultaneous internal stabilization of the r-tuple of systems given by Figure 1 are (C1) the r operator polynomials $D_j o\bar{D}+N_j o\bar{N}$ $j=1,\dots,r$ are stable, i.e.

$$(D_j o\bar{D}+N_j o\bar{N})\cdot y=u \tag{2.12}$$

$j=1,\dots,r$ are simultaneously bounded-input bounded-output (BIBO) stable and (C2) the coefficients $\bar{N}(i,\cdot)$ and $\bar{D}(i,\cdot)$ $i=0,\dots,\bar{q}$ of $\bar{N}$ and $\bar{D}$ which satisfy condition (C1) are uniformly bounded functions of time (i.e. there exists a constant $C<\infty$ such that $\|\bar{N}(i,k)\|<C$ and $\|\bar{D}(i,k)\|<C$ $\forall$ $k\in Z$, $i=0,\dots,\bar{q}$).

In order to satisfy condition (C1) we now define the simultaneous coefficient assignment problem.

Definition 2.5: We say that the r-tuple of systems $D_j^{-1}oN_j$ $j=1,\dots,r$ as defined by (2.5)-(2.6) are simultaneously coefficient assignable by a compensator $\bar{N}o\bar{D}^{-1}$ (as defined by (2.9)-(2.10)) of lag $\bar{q}$ provided that, for a suitable choice of $\bar{N}$ and $\bar{D}$, $\Delta_j(i,k)\in IR^{p\times p}$ $j=1,\dots,r$; $i=1,\dots,\ell+\bar{q}$ defined by

$$(D_j o\bar{D}+N_j o\bar{N})(k) = I + \sum_{i=1}^{\ell+\bar{q}} \Delta_j(i,k)z^{-i} \tag{2.13}$$

may be arbitrarily assigned for $k\geq i$.

The conditions under which a given r-tuple (N_j,D_j) $j=1,\dots,r$ may be simultaneously coefficient assigned are given by the following theorem.

Theorem 2.6 [1],[11]: A generic r-tuple of $p\times m$ systems (2.1) of lag ℓ is simultaneously coefficient assignable by a linear time-varying compensator if

$$m+p>rp \tag{2.14}$$

Provided (2.14) is satisfied, the order of the compensator is the minimum non-negative integer $\bar{q}$ which satisfies the equation

$$\bar{q}[m+p-rp]\geq rp\ell-m \tag{2.15}$$

Proof (Sketch): Equating like powers of z^{-1} in (2.13) for $j=1,\dots,r$, the coefficient assignment problem is solvable precisely when

$$S(k)\nu(k) = \Delta(k) \tag{2.16}$$

has a solution $\nu(k) \forall\ k \geq 0$, where $S(k)$ is a $((\ell+\bar{q})rp+p) \times (\bar{q}+1)(p+m)$ time-varying version of the generalized Sylvester matrix, $\nu(k)$ is a matrix consisting of the coefficients $\bar{N}(k,z^{-1})$, $\bar{D}(k,z^{-1})$ and $\Delta(k)$ is a matrix consisting of the $\Delta_j(i,\cdot)$ $j=1,\ldots,r$; $i=1,\ldots,\ell+\bar{q}$. As (2.14) is satisfied $S(k)$ is generically of full row rank which yields the result.

(Q.E.D.)

Provided the conditions of Theorem 2.6 hold, condition (C1) for the simultaneous internal stabilization of a generic r-tuple of plants holds. The following definition, assuming the conditions of Theorem 2.6 are satisfied, will define the class of systems for which condition (C2) for the simultaneous internal stabilization problem holds as well.

Definition 2.7: The r-tuple $D_j^{-1} o N_j$ $j=1,\ldots,r$ as defined by (2.5)-(2.6) is said to be uniformly bounded coefficient assignable provided that $\forall$ $k \in Z^+$, the set of non-negative integers, there exists a $((\ell+\bar{q})rp+p) \times ((\ell+\bar{q})rp+p)$ minor $S^*(k)$ of $S(k)$, where $S(k)$ is defined by (2.16), which satisfies $|\det S^*(k)| > \varepsilon$ for some $\varepsilon > 0$ which is independent of k.

Combining Proposition 2.3, Theorem 2.6, Equation (2.11) and Definition 2.7 we now give the main result of this section.

Theorem 2.8: Suppose that a given r-tuple of $p \times m$ systems $D_j^{-1} o N_j$ $j=1,\ldots,r$ of lag ℓ are uniformly bounded coefficient assignable. Then the r-tuple of systems are simultaneously internally stabilizable by a time-varying compensator of lag $\bar{q}$ where $\bar{q}$ is the minimum non-negative integer which satisfies $\bar{q}[m+p-rp] \geq rp\ell - m$.

Remark 2.9: It will be noted that the simultaneous stabilization result given by Theorem 2.8 places no apriori upper bound on the rate of variation for the systems parameters.

Remark 2.10: Using the notion of generalized stability regions [6], [7], which shows that if a time-varying system evolves on a sufficiently small neighborhood of a stable time-invariant system the time-varying system is also stable, Theorem 2.8 is robust with respect to parameter uncertainties.

III. EXACT TRACKING FOR LTV SYSTEMS

In this section we are concerned with the ability of a time-varying system to reproduce desired output values. Via similar techniques to those used in considering the simultaneous stabilizability problem, we shall construct a very simple control law which will allow for the stable tracking of a given reference signal $y^*(k)$ (i.e. we can track $y^*(k)$ with a bounded input $u(\cdot)$). It will be noted that i.) our result generalizes results given in [12] for the stable tracking problem for time-invariant systems and ii.) the class of time-varying systems considered are not required to be slowly varying.

We shall consider the class of time-varying systems which can be realized as p×p ARMA models of lag ℓ, i.e. systems of the form

$$D(k,z^{-1})\cdot y(k) = N(k,z^{-1})\cdot u(k) \tag{3.1}$$

where

$$D(k,z^{-1}) = I+\sum_{i=1}^{\ell} D(i,k)z^{-i}; \; N(k,z^{-1})=\sum_{i=0}^{\ell} N(i,k)z^{-i} \tag{3.2}$$

It shall further be assumed that

i.) $y(k)=\underline{0} \;\; \forall k<0$

ii.) $\|D(i,\cdot)\|$ is a uniformly bounded function of time for $i=1,2,\ldots,\ell$

iii.) $N(k,z^{-1})$ is a stable operator and

iv.) $\{y^*(k)\}_{k=0}^{\infty}$ is a bounded sequence in $\mathbb{R}^p$, i.e. $\|y^*(\cdot)\|<M<\infty \;\forall$ $k\in Z$ and $y^*(i)=\underline{0} \;\forall\; i<0$.

Given (3.1) and $\{y^*(k)\}_{k=0}^{\infty}$ define the feedback compensator as follows:

$$[L(k,z^{-1})oN(k,z^{-1})]\cdot u(k) = -[P(k,z^{-1})oN(k,z^{-1})]\cdot y(k) + N(k,z^{-1})\cdot y^*(k) \tag{3.3}$$

where

$$L(k,z^{-1}) = \sum_{i=0}^{q} L(i,k)z^{-i}; \; P(k,z^{-1})=\sum_{i=1}^{q} P(i,k)z^{-i} \tag{3.4}$$

It is easy to show that the closed-loop system (3.1)-(3.4) is given by

$$[L(k,z^{-1})oD(k,z^{-1})+P(k,z^{-1})oN(k,z^{-1})] \cdot y(k) = N(k,z^{-1})\cdot y^*(k) \tag{3.5}$$

Hence if we solve for $L(k,z^{-1})$ and $P(k,z^{-1})$ such that

$$L(k,z^{-1})oD(k,z^{-1})+P(k,z^{-1})oN(k,z^{-1}) = N(k,z^{-1}) \tag{3.6}$$

$\forall\; k\geq 0$ (3.5) becomes

$$N(k,z^{-1})\cdot y(k) = N(k,z^{-1})\cdot y^*(k) \tag{3.7}$$

$\forall$ $k \geq 0$ which implies i.) $y(k)=y^*(k)$ $\forall$ $k \in Z$ and hence ii.) $D(k,z^{-1}) \cdot y^*(k) = N(k,z^{-1}) \cdot u(k)$ which implies via ii.)-iv.) that $\{u(k)\}_{k=0}^{\infty}$ is bounded.

It can be shown, using techniques similar to those used in the previous section, that (3.6) can be solved generically relative to the space of systems (3.1) for $q=\ell$. Thus we have the following theorem.

Theorem 3.1: Given a generic $p \times p$ ARMA system (3.1) which satisfies i.)-iv.) there exists a feedback compensator (3.3) with $q=\ell$ which will stably track any bounded reference signal $\{y^*(k)\}_{k=0}^{\infty}$.

Remark 3.2: It can be shown that even if we do not have perfect information about the plant parameters, the tracking error $\|y(k)-y^*(k)\|$ can be made small if we have "sufficiently good" information about the plant parameters. Similar results may be obtained when $y^*(k)$ is generated by a reference model.

IV. CONCLUSION

In this paper we have given conditions under which a r-tuple of time-varying systems may be robustly internally stabilized. It was noted that our technique is free of the ubiquitious requirement that the time-varying systems considered be slowly varying. Also, a compensator was derived which allows a class of time-varying systems to stably track a given reference signal exactly. Work is currently progressing to extend the class of systems for which such exact stable tracking is possible.

REFERENCES

[1] P.R. Bouthellier, Ph.D. Dissertation, Washington University in St. Louis, in preparation.

[2] B.K. Ghosh, "An Approach to Simultaneous System Design, I. Semialgebraic Geometric Methods," SIAM J. Control and Optimization, Vol. 24, No. 3, May 1986, pp. 480-496.

[3] B.K. Ghosh, "An Approach to Simultaneous System Design, II. Nonswitching Gain and Dynamic Feedback Compensation by Algebraic Geometric Methods," SIAM J. Control and Optimization, Vol. 26, No. 4, July 1988, pp. 919-963.

[4] B.K. Ghosh, "Transcendental and Interpolation Methods in Simultaneous Stabilization and Simultaneous Partial Pole Placement Problems," SIAM J. Control and Optimization, Vol. 24, No. 4, July 1986, pp. 1091-1109.

[5] B.K. Ghosh and C.I. Byrnes, "Simultaneous Stabilization and Simultaneous Pole-Placement by Non-Switching Dynamic Compensation," IEEE Trans. on Automatic Control, Vol. AC-28, No. 6, June 1983, pp. 735-741.

[6] P.R. Bouthellier and B.K. Ghosh, "A Stability Theory of Linear Time-Varying Systems," Proceedings 25th Annual Allerton Conference on Communication, Control and Computing, 1987, pp. 1172-1180.

[7] C.A. Desoer, "Slowly Varying Discrete System $x_{i+1}=A_i x_i$," Electronics Letters, Vol. 6, No. 11, pp. 339-340, 1970.

[8] I.M. Singer and J.A. Thorpe, Lecture Notes on Elementary Topology and Geometry, New York: Scott, Foreman, Glenview, IL, 1967.

[9] E.W. Kamen, P.P. Khargonekar and K.R. Poolla, "A Transfer Function Approach to Linear Time-Varying Discrete-Time Systems," SIAM J. Control and Optimization, Vol. 23, No. 4, July 1985, pp. 550-565.

[10] K.R. Poolla, "Linear Time-Varying Systems; Representation and Control via Transfer Function Matrices," Ph.D. Dissertation, University of Florida, Gainsville, 1984.

[11] P. Bouthellier and B.K. Ghosh, "Robust Stabilization of Discrete-Time Single-Input Single-Output Linear Time-Varying Systems," Proc. of 27th IEEE Conf. on Dec. and Control, 1988, Austin, Texas, 39-44.

[12] G.C. Goodwin and K.S. Sin, Adaptive Filtering Prediction and Control, Prentice-Hall, Englewood Cliffs, NJ, 1984.

INDIRECT REPETITIVE CONTROL FOR LINEAR DISCRETE MULTIVARIABLE SYSTEMS [1]

MINH PHAN [2]
NASA Langley Research Center, Hampton, VA

RICHARD W. LONGMAN [3]
Columbia University, New York, NY

JER-NAN JUANG [4]
NASA Langley Research Center, Hampton, VA

This paper presents an extension of a learning control formulation to the repetitive control problem using the concept of a repetition domain. The formulation is applicable to general discrete linear multivariable systems with constant or periodic coefficents, and with a repetitive disturbance possibly present. The method is indirect in that it involves estimation of the system repetition domain parameters as an intermediate step in computing the learning control input. Within this mathematical framework, adaptive techniques are used to develop learning control laws. Despite initial lack of knowledge of the system parameters, stability of the learning process is guaranteed.

INTRODUCTION

A large fraction of control systems in use execute repetitive operations, e.g., controllers for robots in assembling and manufacturing systems. Recently, two bodies of work have appeared in the literature to develop controllers that learn from previous experience at performing a repetitive command to improve tracking performance. These are known as repetitive control and learning control.

Repetitive control and learning control aim to make explicit use of the knowledge of the repetitive nature of the process and the command. They also aim to correct for the effect of a repetitive time-varying forcing function either in the form of a disturbance or of modelling errors. Such forcing functions are common in repetitive systems. Repetitive control differs from learning control in that it applies to systems which at the end of a repetition of the task, the system need not return to the same initial condition for the next repetition. Learning control, on the other hand, learns from repeating tasks that start from the same initial condition each time. This task dependency is a primary distinguishing feature between repetitive control and learning control.

Some recent results in repetitive control are listed in Refs.[1-4] and in a series of papers by Tomizuka et al., for example, Refs.[5,6]. References [7,8] and the references listed therein present some basic results in learning control. Recent advances have been made possible by a general mathematical formulation of the problem in the repetition domain, Refs.[9-12]. It is the purpose of this paper to extend this formulation to the repetitive control problem. This will then provide one unified mathematical framework for both classes of problems, which until now remained separate in formulations, yet related in concepts.

MATHEMATICAL FORMULATION

Consider a general discrete multivariable linear system, which at the present for clarity is taken to be time-invariant

$$\begin{aligned} x(i+1) &= Ax(i) + Bu(i) + w(i) \\ y(i) &= Cx(i) \end{aligned} \tag{1}$$

---

[1] Research supported by a National Research Council Research Associateship awarded to the first author.

[2] NRC Research Associate, Spacecraft Dynamics Branch.

[3] Professor of Mechanical Engineering.

[4] Principal Scientist, Spacecraft Dynamics Branch.

where $x \in R^n$, $y \in R^q$, $u \in R^m$, $w(i)$ denotes a periodic disturbance of period p time steps. A, B, C and $w(i)$ are asummed to be unknown. For $i = 1, 2, \ldots, p$, the solution to (1) is

$$x(i) = A^i x(0) + \sum_{\tau=0}^{i-1} A^{i-\tau-1} B u(\tau) + \sum_{\tau=0}^{i-1} A^{i-\tau-1} w(\tau) \tag{2}$$

For $i = p, p+1, \ldots, 2p$, we write the solution to (1) as

$$x(i) = A^{i-p} x(p) + \sum_{\tau=0}^{i-(p+1)} A^{i-\tau-(p+1)} B u(\tau+p) + \sum_{\tau=0}^{i-(p+1)} A^{i-\tau-(p+1)} w(\tau+p) \tag{3}$$

We introduce a repetition variable j, and a new time step variable k, so that the general solution to (1) for the first, second, and all subsequent periods can be expressed as

$$x(i) = A^{i-jp} x(jp) + \sum_{\tau=0}^{i-(jp+1)} A^{i-\tau-(jp+1)} B u(\tau+jp) + \sum_{\tau=0}^{i-(jp+1)} A^{i-\tau-(jp+1)} w(\tau+jp) \tag{4}$$

for $i = jp + k$, $j = 0, 1, 2, \ldots$, $k = 1, 2, \ldots, p$. At any time step i, the state vector $x(i)$ is written as

$$x(jp+k) = x_j(k) \quad and \quad x(jp) = x_j(0) \tag{5}$$

where $k = i - jp$, and similarly for $y(i)$. Associated with each state vector $x(i)$ is an input vector $u(i-1)$ which is also rewritten as

$$u(i-1) = u_j(k-1) \tag{6}$$

Next, we define a difference operator in the newly defined variable j such that

$$\delta_j z(k) = z_j(k) - z_{j-1}(k) \tag{7}$$

Now applying the operator (7) to both sides of equation (4) after making all the changes of variables from i to j and k, and noting that $\delta_j w(k) = 0$ since $w(i)$ is periodic of period p, we obtain the following description of the repetitive process in the repetition domain

$$\delta_{j+1} \underline{x} = \underline{A}\, \delta_{j+1} x(0) + P\, \delta_{j+1} \underline{u} \tag{8}$$

where

$$\underline{x} = \left[x^T(1) \;\; x^T(2) \;\; \ldots \;\; x^T(p) \right]^T \qquad \underline{u} = \left[u^T(0) \;\; u^T(1) \;\; \ldots \;\; u^T(p-1) \right]^T$$

$$\underline{A} = \left[A^T \; \left(A^2\right)^T \; \ldots \; \left(A^p\right)^T \right]^T$$

$$P = \begin{bmatrix} B & & & \mathbf{O} \\ AB & B & & \\ \vdots & \vdots & \ddots & \\ A^{p-1}B & A^{p-2}B & \cdots & B \end{bmatrix}$$

Using $y_j(k) = Cx_j(k)$, the equivalent output description to (8) above is

$$\delta_{j+1} \underline{y} = \underline{CA}\, \delta_{j+1} x(0) + P_0 \delta_{j+1} \underline{u} \tag{9}$$

where

$$\underline{y} = \left[y^T(1)\, y^T(2) \ldots y^T(p) \right]^T$$

$$\underline{C} = diag\left[C \;\; C \;\; \ldots \;\; C \right] \qquad P_0 = \underline{C}\; P$$

If C is square and full rank then duality in forms of the state description and output description of the repetitive system in the repetition domain is obtained by noting

$$\delta_{j+1}\, \underline{y} = \underline{A}_0 \delta_{j+1}\, y(0) \;+\; P_0 \delta_{j+1}\, \underline{u} \tag{10}$$

where $[\underline{y}, \underline{A}_0, y(0), P_0]$ plays the role of $[\underline{x}, \underline{A}, x(0), P\,]$ in (8) respectively, and $\underline{A}_0 = \underline{C}\, \underline{A}\, \underline{C}^{\,-1}$

Note that if the system returns to the same initial condition every repetition, then this is equivalent to setting $\delta_{j+1}\, x(0) = 0$ in (8), (9) (or $\delta_{j+1}\, y(0) = 0$ in (10)). We then have

$$\delta_{j+1}\, \underline{x} \;= P\; \delta_{j+1}\, \underline{u} \tag{11}$$

and similarly for the output description as before in the learning control problem.

Similar to the learning control problem, one must address the issue of trajectory feasibility and its specification; namely existence of an input sequence that will produce the desired trajectory specified in an appropriate manner for the system. For the present development, we will work with equation (10) above for the simple case $q = m = n$. Then any desired output trajectory, denoted by $y^*(i)$, is a feasible trajectory provided that the product CB is full rank. The reader is refered to Refs.[9] and [12] for a discussion of how to handle generalizations to the cases $q > m$ and $n \geq m$. Let the rows of $\underline{A}_0$ and P_0 be defined as

$$a_l(k) = \left[a_{(k\text{-}1)n+l,1} \quad a_{(k\text{-}1)n+l,2} \;\cdots\; a_{(k\text{-}1)n+l,n} \right]^T$$

$$p_l(k) = \left[p_{(k\text{-}1)n+l,1} \quad p_{(k\text{-}1)n+l,2} \;\cdots\; p_{(k\text{-}1)n+l,mk} \right]^T$$

where the subscripts on a and p denote positions of the elements in the matrix $\underline{A}_0$ and P_0, that form the column vectors $a_l(k)$ and $p_l(k)$ respectively, $l = 1, 2, \ldots, q$ and $k = 1, 2, \ldots, p$. The mk-dimensional input variation vector $\delta_{j+1}\underline{u}_m(k\text{-}1)$ is defined as

$$\delta_{j+1}\underline{u}_m(k\text{-}1) = \left[\delta_{j+1}u_1(0) \;\ldots\; \delta_{j+1}u_m(0) \;\ldots\; \delta_{j+1}u_1(k\text{-}1) \;\cdots\; \delta_{j+1}u_m(k\text{-}1) \right]^T$$

Making use of the above definitions, (10) can then be expressed as

$$\delta_{j+1}y_l(k) = a_l^T(k)\delta_{j+1}y(0) + p_l^T(k)\delta_{j+1}\underline{u}_m(k\text{-}1) \tag{12}$$

for all (l,k) pairs. Equation (12) represents a collection of single-output, multiple-input models in the repetition domain, with input vectors of growing dimensions. Please note that even though the actual system in the time domain may be multiple-input, multiple-output, in the repetition domain it takes the form of a set of single-input, multiple-output ARMA-like models. Equation (12) can be rewritten in simpler forms by defining augmented parameter and input variation vectors, namely

$$p_l^{+}(k) = \left[a_l^T(k) \quad p_l^T(k) \right]^T$$

$$\delta_{j+1}\underline{u}_m^{+}(k\text{-}1) = \left[\delta_{j+1}^T y(0) \quad \delta_{j+1}^T \underline{u}_m(k\text{-}1) \right]^T$$

then it becomes

$$\delta_{j+1}y_l(k) = \; p_l^{+}(k)^T \delta_{j+1}\underline{u}_m^{+}(k\text{-}1) \tag{13}$$

for all (l,k) pairs, $l = 1, 2, \ldots, q;\; k = 1, 2, \ldots, p$.

REPETITIVE CONTROL SCHEME

There are various adaptive control schemes that can be applied to the system in (13) to generate repetitive control laws. For clarity of exposition, we derive a simple scheme that is based on the projection algorithm, Ref.[13]. Other schemes can be similarly derived, and will be studied later. The parameter estimate of $p_l^+(k)$ at repetition j, denoted by $\hat{p}_{l,j}^+(k)$ can be updated according to the rule

$$\hat{p}^{+}_{l,j}(k) = \hat{p}^{+}_{l,j-1}(k) + a_j \delta_j \underline{u}^{+}_{m}(k\text{-}1)\left[\frac{\delta_j y_l(k) - \delta_j^T \underline{u}^{+}_m(k\text{-}1)\hat{p}^{+}_{l,j-1}(k)}{1 + \delta_j^T \underline{u}^{+}_m(k\text{-}1)\delta_j\underline{u}^{+}_m(k\text{-}1)}\right] \tag{14}$$

for all (l,k) pairs, and $0 < a_j < 2$. The augmented repetitive control variation vector $\delta_{j+1}\underline{u}_m^+(k\text{-}1)$ is required to satisfy

$$\hat{p}^{+}_{l,j}(k)^T \delta_{j+1}\underline{u}^{+}_m(k\text{-}1) = y^*_{l,j+1}(k) - y_{l,j}(k) \tag{15}$$

for all (l,k) where $y^*_{l,j+1}(k)$ denotes the desired value of the l-th output of $y_{j+1}(k)$ at the time step k of repetition $j+1$. The repetitive control variation vector $\delta_{j+1}\underline{u}_m(k\text{-}1)$ is then computed as the following so that (15) is satisfied

$$\hat{p}_{l,j}(k)^T \delta_{j+1}\underline{u}_m(k\text{-}1) = y^*_{l,j+1}(k) - y_{l,j}(k) - \hat{a}^T_{l,j}(k)\delta_{j+1}y(0) \tag{16}$$

The repetitive control input itself to be applied to the system is computed from

$$\underline{u}_{j+1} = \underline{u}_j + \delta_{j+1}\underline{u} \tag{17}$$

where $\delta_{j+1}\underline{u}$ is simply $\delta_{j+1}\underline{u}_m(k\text{-}1)$ when $k = p$. Furthermore, the 1 in the denominator of the term in the square bracket in equation (14) is intended to prevent division by zero, in practice any positive number can be used.

STABILITY

If the desired trajectory is feasible, i.e. if there exists a control which causes the system to produce the desired trajectory, then the tracking error of the repetitive control scheme (14)-(17) converges to zero as the repetitions progress.

To show tracking covergence in the time domain, it is equivalent to show that in the repetition domain

$$\lim_{j \to \infty} [y_{l,j}(k) - y^*_{l,j}(k)] = 0 \quad ; \quad k = 1, 2, ..., p \tag{18}$$

A complete proof follows four basic steps outlined below.

Step 1: The Euclidean norm of the parameter estimation error defined by

$$\tilde{p}^{+}_{l,j}(k) = p^{+}_{l}(k) - \hat{p}^{+}_{l,j}(k)$$

for the learning (repetitive) control algorithm (14)-(17) in the repetition domain satisfies

$$\| \tilde{p}^{+}_{l,j}(k) \| - \| \tilde{p}^{+}_{l,j-1}(k) \| \le 0 \tag{19}$$

for all (l,k) pairs.

Step 2: Define the tracking error as

$$e_{l,j}(k) = y^*_{l,j}(k) - y_{l,j}(k)$$

The learning control algorithm (14)-(17) applied to the system given in the repetition domain by a set of ARMA models in (13) yields

$$\lim_{j \to \infty} \frac{e_{l,j}^{2}(k)}{1 + \delta_j^{T} \underline{u}_m^{+}(p\text{-}1)\delta_j \underline{u}_m^{+}(p\text{-}1)} = 0 \tag{20}$$

Step 3: In order to show (20) implies that the tracking error itself, $e_{l,j}(k)$, approaches zero, it is necessary that the denominator be bounded. However, it is sufficient to show that $\|\delta_j \underline{u}_m^{+}(p\text{-}1)\|$ is bounded with respect to $|e_{l,j}(k)|$ in some fashion.

Step 4: In particular, if the following boundedness condition holds

$$\|\delta_j \underline{u}_m^{+}(p\text{-}1)\| \le C_1 + C_2 \underset{0 \le \zeta \le j}{Max} |e_{l,\zeta}(k)| \tag{21}$$

for some $0 \le C_1 < \infty$, $0 < C_2 < \infty$ then (21) implies

a) The sequence $\|\delta_j \underline{u}_m^{+}(p\text{-}1)\|$ is bounded, and

b) $e_{l,j}(k)$ approaches zero as j approaches infinity.

The above is a standard technical lemma of adaptive control theory when specialized to the present problem, Ref.[13].

Step 4: In order to establish (21), one needs to go through some lengthy arguments which can be outlined here as follows: If the system in (1) is such that given a bounded desired output trajectory, a bounded input history exists to produce it, i.e., the feasibility condition on the desired trajectory is met, then bounded output $|y_l(i)|$ implies bounded input $\|u(i)\|$. For bounded desired trajectory the output error $|e_l(i)|$ is bounded. Bounded $\|u(i)\|$ also implies bounded $\|\delta_j u(k)\|$, and hence bounded $\|\delta_j \underline{u}_m(k\text{-}1)\|$ and consequently $\|\delta_j \underline{u}_m^{+}(p\text{-}1)\|$. One can then show that (21) holds for some C_1, C_2 and the tracking objective is achieved. However, the above argument assumes that the output trajectory, and hence the output error is bounded. It can be shown that this indeed must be the case since unboundedness of the output trajectory will lead to a contradiction. This will then complete the proof.

COMPUTATION PROCEDURE

Step 1: Apply an arbitrarily chosen input sequence $\underline{u}_0$ to the system, measure $\underline{y}_0$, $y_1(0)$.

Step 2: Construct initial guesses of P_0, $\underline{A}_0$ from initial guesses of $a_{l,j}(k)$ and $p_{l,j}(k)$ for $j = 0$. Then compute from (16)

$$\delta_1 \underline{u} = \widehat{P}_{0,0}^{-1}[\delta_1^{*} \underline{y} - \widehat{A}_{0,0}\delta_1 y(0)]$$

$$\delta_1^{*} \underline{y} = \underline{y}_1^{*} - \underline{y}_0 \quad , \quad \delta_1 y(0) = y_1(0) - y_0(0) \quad , \quad y_0(0) = y(0)$$

Step 3: Compute the control sequence to be applied at the first repetition $\underline{u}_1 = \underline{u}_0 + \delta_1 \underline{u}$, apply $\underline{u}_1$ to the system, measure $\underline{y}_1$, $y(2p) = y_1(p) = y_2(0)$.

Step 4: Update $\hat{P}_{0,0}$, $\hat{\underline{A}}_{0,0}$ to obtain $\hat{P}_{0,1}$, $\hat{\underline{A}}_{0,1}$ by (14)

$$\widehat{p}_{l,1}^{+}(k) = \widehat{p}_{l,0}^{+}(k) + a_1 \delta_1 \underline{u}_m^{+}(k\text{-}1)\left[\frac{\delta_1 y_l(k) - \delta_1^{T} \underline{u}_m^{+}(k\text{-}1)\widehat{p}_{l,0}^{+}(k)}{1 + \delta_1^{T} \underline{u}_m^{+}(k\text{-}1)\delta_1 \underline{u}_m^{+}(k\text{-}1)}\right]$$

$$p_{l,0}^{+}(k) = \left[a_{l,0}^{T}(k) \quad p_{l,0}^{T}(k)\right]^{T} \qquad \delta_1 \underline{u}_m^{+}(k\text{-}1) = \left[\delta_1^{T} y(0) \quad \delta_1^{T} \underline{u}_m(k\text{-}1)\right]^{T}$$

$$\delta_1 y_l(k) = y_{l,1}(k) - y_{l,0}(k)$$

Step 5: After obtaining $\hat{P}_{0,1}$, $\hat{\underline{A}}_{0,1}$ then compute

$$\delta_2\underline{u} = \widehat{P}_{0,1}^{-1}[\delta_2^*\underline{y} - \widehat{A}_{0,1}\delta_2 y(0)]$$

$$\delta_2^*\underline{y} = \underline{y}_2^* - \underline{y}_1 \qquad \delta_2 y(0) = y_2(0) - y_1(0)$$

Step 6: Apply $\underline{u}_2 = \underline{u}_1 + \delta_2\underline{u}$ to the system, then measure $\underline{y}_2$; $y(3p) = y_2(p) = y_3(0)$.

Step 7: Repeat the cycle from *Step 4* by updating $\widehat{P}_{0,1}, \underline{\widehat{A}}_{0,1}$ to obtain $\widehat{P}_{0,2}, \underline{\widehat{A}}_{0,2}$ etc...

REMARKS

Remark 1: We note here some immediate results derived from the system description in the repetition domain given in equation (8), (9), or (10) with regard to the nature of the control input necessary to track a periodic trajectory. Consider, say the state/input description given in (8) with $\delta_{j+1}\underline{u}$ serving as the control input in the repetition domain. Note that $\delta_{j+1}x(0)$ is the same as $\delta_j x(p)$. The homogeneous equation (denoted with the subscript h) corresponding to $\delta_{j+1}\underline{u} = 0$ is then given as

$$\delta_{j+1}\underline{x}_h - \underline{A}\ \delta_j x_h(p) = 0 \tag{22}$$

$$\underline{x}_h = [\ x_h^T(1)\ x_h^T(2)\ \ \ldots\ \ x_h^T(p)\ ]^T$$

We can now make the following comments:

a) The homogeneous equation of the system in the repetition domain given in (22) characterizes the response of the system (1) in the time domain under a periodic forcing function (or input) $u(i)$ of period of p time steps.

b) The steady state response under periodic input can then be characterized solely in terms of the A matrix. Specifically, if A is asymptotically stable, then under a periodic input, the steady state response is periodic.

c) If A is unstable, or marginally stable (i.e., having at least one eigenvalue on the unit circle in the complex plane) then (22) cannot be asymptotically stable. Nevertheless, (22) contains a periodic solution.

d) All comments mentioned in the above are independent of any periodic disturbance of period p.

To show *a)* we note that when the forcing term in (8) is set to zero, i.e. $\delta_{j+1}\underline{u} = 0$ then $\underline{u}_j$ is constant. Constant $\underline{u}_j$ in the repetition domain implies that $u(i)$ is periodic in the time domain with period p and the result follows. To show *b)* we need to solve equation (22) for $\underline{x}_h$ when $\delta_j x(p)$ is not identically zero. This can be done by noting that from (22)

$$\delta_{j+1}x_h(p) - A^p\ \delta_j x_h(p) = 0 \tag{23}$$

For simplicity, we will show for the case A is diagonalizable, $A = M^{-1}\Lambda M$. We define $z_j(p) = M^{-1}x_j(p)$, (23) becomes

$$z_{j+1}(p) - (I + \Lambda^p)z_j(p) + \Lambda^p z_{j-1}(p) = 0 \tag{24}$$

whose steady state behavior we want to investigate. If A is asymptotically stable, then as j approaches infinity, $z_j(p)$ approaches some constant value. Hence,

$$\lim_{j\to\infty}\ \delta_j x(p) = 0$$

It will then follow from (23) that

$$\lim_{j\to\infty}\ \delta_{j+1}x(1)\ =\ \lim_{j\to\infty}\ \delta_{j+1}x(2) =\ \ldots\ =\ \lim_{j\to\infty}\ \delta_{j+1}x(p\text{-}1)\ = 0$$

Thus $\delta_{j+1}\underline{x} = 0$ in the limit as j approaches infinity, or the steady state response is periodic. To show *c)* consider the case where A is only marginally stable or unstable. Let λ_i , $i = 1, 2, ..., n$ denote the eigenvalues of A. Equation (24) is a set of n decoupled characteristic equations. If any of these corresponds to an unstable system then the set is unstable. The roots of the i-th characteristic equation are 1 and $(\lambda_i)^p$. If $|\lambda_i| \leq 1$ and there is at least one root whose magnitude is one, possibly repeated, then if among the set of roots whose magnitude is one contains

i) at least one root $\lambda_i = 1$, then the system is unstable.
ii) at least one root $\lambda_i = -1$ and p is even then the system is unstable.
iii) otherwise, the system is maginally stable.

Finally to show *d),* we simply note that (22) is independent of the repetitive disturbance $w(i)$ of period p time steps.

The comments discussed here are by no means surprising. The interesting fact is that they are derived from the repetition domain framework. They provide insights as to what the system description in the repetition domain means, i.e., what it means to view the system from the repetition domain, an approach that so far has proved itself quite useful.

Remark 2: For clarity, we have asummed that the system is time-invariant, but generalizations to time-varying systems with periodic coefficients, i.e. repeating with each repetition can be similarly derived. The system in (1) becomes

$$\begin{aligned} x(i+1) &= A(i)x(i) + B(i)u(i) + w(i) \\ y(i) &= C(i)x(i) \end{aligned}$$

where $A(i) = A(i+jp)$, $B(i) = B(i+jp)$ for $i = 0, 1, ..., p\text{-}1$ and $j = 0, 1, 2, ...$

Remark 3: The repetition domain description converts the tracking problem of a periodic trajectory in the time domain to a regulation problem in the repetion domain. Repetitive control is generally designed to track a periodic trajectory. However, for the learning scheme presented here the desired trajectory need not be periodic. i.e., the desired trajectory can be changing every repetition. This extends the learning idea to the tracking of a general time-varying trajectory for systems which can contain periodic coefficients and periodic disturbances. This corresponds to the general tracking problem in the repetition domain.

Remark 4: In general p corresponds to the period of the desired trajectory and/or the repetitive disturbance in the repetitive system. A special case of *Remark 3* is $p = 1$. All of the above results when specialized to this case apply to the tracking problem of a general desired trajectory, periodic or non-periodic, for a time-invariant system with a constant disturbance possibly present.

Remark 5: The basic control law presented here if applied directly to equation (9) require state measurements at every p steps, e.g. $x(0)$, $x(p)$, $x(2p)$, etc... , in addition to the output measurements. The need for state measurements can be avoided if the following modifications to the basic algorithm are made. Let the rows of $\underline{C}\,\underline{A}$ in (9) be defined as

$$c_l(k) = \left[c_{(k\text{-}1)q+l,1} \quad c_{(k\text{-}1)q+l,2} \quad \cdots \quad c_{(k\text{-}1)q+l,n} \right]^T$$

then (9) can be written as a set of ARMA-like models as

$$\begin{aligned} \delta_{j+1}y_l(k) &= c_l^T(k)\delta_{j+1}x(0) + p_l^T(k)\delta_{j+1}\underline{u}_m(k\text{-}1) \\ &= p_l^{+}(k)^T\ \delta_{j+1}\underline{u}_m^{+}(k\text{-}1)' \end{aligned} \tag{25}$$

where

$$p_l^{+}(k)' = \left[p_l(k)_0 \quad p_l^T(k)\right]^T \; ; \; p_l(k)_0 = c_l^T(k)\delta_{j+1}x(0)$$

$$\delta_{j+1}\underline{u}^{+}(k\text{-}1)' = \left[1 \quad \delta_{j+1}^T\underline{u}_m(k\text{-}1)\right]^T$$

for all (l,k) pairs, $l = 1, 2, ..., q$; $k = 1, 2, ..., p$. The parameter update equation now becomes

$$\widehat{p}_{l,j}^{+}(k)' = \widehat{p}_{l,j\text{-}1}^{+}(k)' + a_j\delta_j\underline{u}_m^{+}(k\text{-}1)'\left[\frac{\delta_j y_l(k) - \delta_j^T\,\underline{u}_m^{+}(k\text{-}1)'\,\widehat{p}_{l,j\text{-}1}^{+}(k)'}{1 + \delta_j^T\,\underline{u}_m^{+}(k\text{-}1)'\,\delta_j\underline{u}_m^{+}(k\text{-}1)'}\right] \qquad (26)$$

The new augmented repetitive control variation vector $\delta_{j+1}\underline{u}_m{}^{+}(k\text{-}1)'$ is required to satisfy

$$\widehat{p}_{l,j}^{+}(k)'^{\,T}\delta_{j+1}u_m^{+}(k\text{-}1)' = y_{l,j+1}^{*}(k) - y_{l,j}(k) \qquad (27)$$

From (27) the repetitive control variation vector $\delta_{j+1}\underline{u}_m(k\text{-}1)$ is then computed as

$$\widehat{p}_{l,j}(k)^T\delta_{j+1}\underline{u}_m(k\text{-}1) = y_{l,j+1}^{*}(k) - y_{l,j}(k) - \widehat{p}_{l,j}(k)_0 \qquad (28)$$

Remark 6: The repetitive control scheme presented here is based on the projection algorithm. Use of other parameter estimation algorithms is also possible, and stability analysis can be done analogously. For example, if one uses the least square algorithm for parameter estimation then (14) of the basic algorithm is replaced by

$$\widehat{p}_{l,j}^{+}(k) = \widehat{p}_{l,j\text{-}1}^{+}(k) + R_{j\text{-}2}(k)\delta_j\underline{u}_m^{+}(k\text{-}1)\left[\frac{\delta_j y_l(k) - \delta_j^T\,\underline{u}_m^{+}(k\text{-}1)\widehat{p}_{l,j\text{-}1}^{+}(k)}{1 + \delta_j^T\,\underline{u}_m^{+}(k\text{-}1)R_{j\text{-}2}(k)\delta_j\underline{u}_m^{+}(k\text{-}1)}\right] \qquad (29)$$

where $R_{j\text{-}2}(k)$ is updated according to the rule

$$R_{j\text{-}1}(k) = R_{j\text{-}2}(k) - \left[\frac{R_{j\text{-}2}(k)\delta_j\underline{u}_m^{+}(k\text{-}1)\delta_j^T\,\underline{u}_m^{+}(k\text{-}1)R_{j\text{-}2}(k)}{1 + \delta_j^T\,\underline{u}_m^{+}(k\text{-}1)R_{j\text{-}2}(k)\delta_j\underline{u}_m^{+}(k\text{-}1)}\right]$$

$R_{\text{-}1}(k) = \alpha I_{mk \times mk}$, $\alpha > 0$, and the rest of the algorithm remains the same. The least square algorithm that replaces (26) is similar.

Remark 7: So far we have treated $u(i)$ in (1) as the learning control input. If linear feedback, characterized by the gain F, is already present in the system, we let $u(i) = Fy(i) + v(i) = FCx(i) + v(i)$, where $v(i)$ is the additive learning control input to the system. When substitued into the state equation (1), $[A+BFC]$ and $v(i)$ play the same roles as A and $u(i)$. Therefore, treating $u(i)$ in (1) as simply the learning control input represents no loss of generality.

EXAMPLES

In this section we provide some simple numerical examples to illustrate the working of the learning control presented here.

Example 1: The repetitive learning scheme is applied to a second-order single-input, single-output system.

The system matrices in (1) used in this example are

$$A = \begin{bmatrix} 0.75 & 0 \\ 0.5 & 0.5 \end{bmatrix} \quad B = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad C = [1 \;\; 1]$$

The repetitive disturbance is

$$w(i) = \begin{bmatrix} -0.5sin(2\pi(i\text{-}1)/p) \\ 0.5sin(4\pi(i\text{-}1)/p) \end{bmatrix}$$

The period of the system and the disturbance is p = 10, the initial state $x(0) = [1 \;\; 0]^T$. The desired trajectory is chosen to be $y^*(i) = sin(2\pi i/p)$. The initial trial is generated by an arbitrarily chosen input sequence. The first repetition input is computed from (16) with arbitrarily guessed values of $P_0 = 2I$ and $\underline{A}_0 = [0 \;\; 0]$. The learning algorithm starts at the second repetition, i = $2p$ = 20, and a_j is chosen to be $a_j = (1 - 0.7^j)$ so that corrections to the parameter estimates, and hence to the control signal are relatively small in the first few repetitions. First, when there is no repetitive disturbance present, $w(i) = [0 \;\; 0]^T$, the system response is shown in Fig.1. With the repetitive disturbance present, Fig.2 shows that the tracking objective is achieved with the same control law. In these two cases, in addition to the output measurements, we assume that the state measurements at $i = p, 2p, 3p, \ldots$ are known.

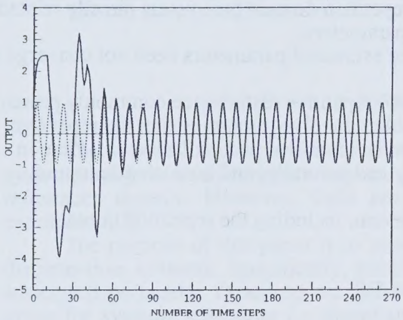


Fig.1 Tracking convergence with no repetitive disturbance present (Example 1)

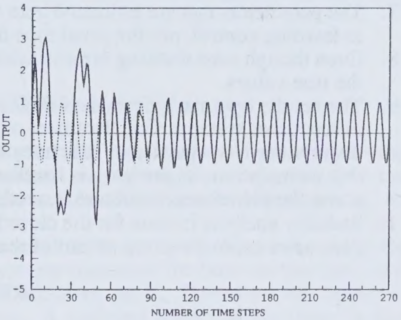


Fig.2 Tracking convergence with repetitive disturbance present (Example 1)

Example 2: Specialize the results for the case p = 1. As mentioned in *Remark 4,* this corresponds to the general tracking solution under a constant disturbance, $w(i) = [0.5 \;\; 0.5]^T$. The result is shown in Fig.3. Note that this is also an extreme case that illustrates that the desired trajectory need not be periodic, i.e. it can change with every repetition.

Example 3: Consider the system in *Example 1*, but now the state measurements are assumed not available and the scheme is based only on output measurements. The modification proposed in *Remark 5* is used here, and the response of the system with the repetitive disturbance present is shown in Fig.4

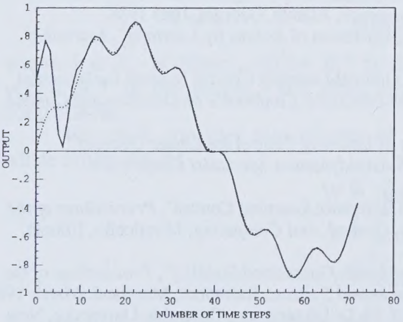


Fig.3 Tracking of a general time-varying trajectory under a constant disturbance Special case p=1. (Example 2)

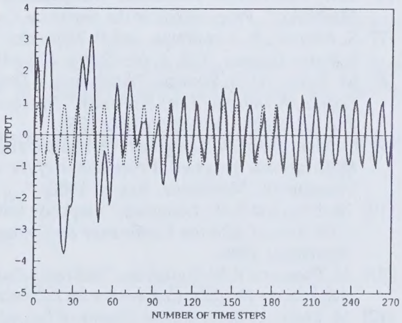


Fig.4 System response using modified algorithm without state measurements (Example 3)

CONCLUSIONS

In conclusion, the basic framework presented in this paper has the following features:

1. It extends the learning control formulation in Ref.[11] to the repetitive control problem employing the concept of a repetition domain.
2. Within this framework, adaptive techniques are used to make the system learn to track a given desired trajectory, which need not be periodic.
3. The formulation assumes no prior knowledge of the system parameters.
4. Convergence of the learning process to zero tracking error is guaranteed provided the system can perform the desired trajectory.
5. The system parameters may be time-invariant or time-varying, but periodic, i.e. repeating with each repetition.
6. The approach employed here is indirect in the sense that it involves on-line parameter estimation as an intermediate step in order to compute the control sequence. Direct methods will be investigated later.
7. The parameters that are estimated here are the repetition domain parameters directly related to learning control, not the usual time domain parameters.
8. Even though zero tracking error is achieved, the estimated parameters need not converge to the true values.
9. The results here are independent of any unknown repetitive disturbance commonly present in repetitive systems. One treatment of this problem involves assuming that the repetitive disturbance has a particular structure, which can be identified and cancelled out. Here in this formulation, its presense is handled simply and naturally, and no such assumption about the disturbance structure is needed.
10. Stability analysis is done for the closed loop system, including the repetition domain parameter estimation step as part of the loop.

REFERENCES

[1] T. Mita and E. Kato, "Iterative Control and Its Applications to Motion Control of Robot Arm -- A Direct Approach to Servo Problems", *Proceedings of the 24th IEEE Conference on Decision and Control,* Fort Lauderdale, Fla., December 1985.

[2] K. Furuta and M. Yamakita, "Iterative Generation of Optimal Input of a Manipulator", *Proceedings of the IEEE Conference on Robotics,* 1986.

[3] R.H. Middleton, G.C. Goodwin and R.W. Longman, "A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task", Technical Report EE 8546, Department of Electrical and Computer Engineering, University of Newcastle, Newcastle, New South Wales, Australia, December 1985.

[4] S. Hara, T. Omata, and M . Nakano, "Synthesis of Repetitive Control Systems and Its Applications", *Proceedings of the 24th IEEE Conference on Decision and Control,* Fort Lauderdale, Fla. December 1985.

[5] M. Tomizuka, T.-C Tsao, and K.-K Chew, "Discrete-time Domain Analysis and Synthesis of Repetitive Controllers", *Proceedings of the American Control Conference,* Atlanta, Georgia, June 1988.

[6] T.-C. Tsao and M. Tomizuka, "Adaptive and Repetitive Digital Control Algorithms for Noncircular Machining", *Proceedings of the American Control Conference,* Atlanta, Georgia, June 1988.

[7] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering Operation of Robots by Learning", *Journal of Robotic Systems,* Vol. 1, No. 2, pp. 123-140, 1984.

[8] M. Togai and O. Yamano, "Analysis and Design of an Optimal Learning Control Scheme for Industrial Robots: A Discrete-Time Approach", *Proceedings of the 24th IEEE Conference on Decision and Control,* Fort Lauderdale, Fla., December 1985.

[9] M. Phan and R.W. Longman, "A Mathematical Theory of Learning Control for Linear Discrete Multivariable Systems", *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference,* Minneapolis, Minnesota, August 1988.

[10] M. Phan and R.W. Longman, "Liapunov Based Model Reference Learning Control", *Proceedings of the 26th Annual Allerton Conference on Communication, Control, and Computing,* Monticello, Illinois, September 1988.

[11] M. Phan and R.W. Longman, "Indirect Learning Control with Guaranteed Stability", *Proceedings of the 1989 Johns Hopkins Conference on Information Sciences and Systems,* Baltimore, Maryland, March 1989.

[12] M. Phan, *A Mathematical Theory of Learning Control,* Ph.D. Dissertation, Columbia University, New York, New York, May 1989.

[13] G.C. Goodwin and K.S. Sin, *Adaptive Filtering, Prediction, and Control,* Prentice Hall, Englewood Cliffs, New Jersey, 1984.

ROBUSTNESS OF COVARIANCE CONTROLLERS

G. ZHU, M. CORLESS, R. SKELTON

School of Aeronautics and Astronautics
Purdue University
West Lafayette, Indiana

Abstract

This paper introduces several different new robustness results for discrete-time systems. Specifically, stability robustness measures for both unstructured and structured parameter variations are presented. In addition, an L_∞ bound on the output is given for systems subject to L_2 disturbance inputs. A unifying aspect of the paper is that all the robustness measures presented are expressed in terms of the system state covariance matrix.

1. INTRODUCTION

The recent literature contains many results on the robustness of continuous-time systems subject to disturbance inputs and parameter variations; see, e.g., [3] and the references therein. However, there are very few results for discrete-time systems; an example is [2]

The purpose of this paper is to introduce several different new robustness results for discrete-time systems. Specifically, stability robustness measures for both unstructured and structured parameter variations are presented. In addition, an L_∞ bound on the output is given for systems subject to L_2 disturbance inputs. A unifying aspect of the paper is that all the robustness measures presented are expressed in terms of the system state covariance matrix. Hence we describe here the robustness properties of any controller (such as a covariance controller [1]) that gives a specified state covariance to the closed loop system.

The paper is outlined as follows. Section 2 derives relationships between disturbance rejection and covariance properties. Section 3 derives relationships between structured and unstructured parameter variations and covariance properties. Also Sections 2 and 3 give examples illustrating the use of the theorems.

2. DISTURBANCE REJECTION AND COVARIANCES

Consider a discrete-time system described by

$$\begin{aligned} x(k+1) &= Ax(k) + Dw(k) \\ y(k) &= Cx(k) \end{aligned} \tag{2.1}$$

where $k \in Z_+$ is "time", $x(k) \in \mathbb{R}^n$ is the state, $w(k) \in \mathbb{R}^{n_w}$ is a disturbance input, and $y(k) \in \mathbb{R}^{n_y}$ is the output. We shall suppose A is asymptotically stable and (A, D) is controllable.

We shall consider two classes of disturbances for $w(\cdot)$. When $w(\cdot)$ is zero mean white noise, we let

$$\begin{aligned} W &\triangleq \lim_{k\to\infty} E[w(k)w^*(k)] \\ X &\triangleq \lim_{k\to\infty} E[x(k)x^*(k)] \\ Y &\triangleq \lim_{k\to\infty} E[y(k)y^*(k)] . \end{aligned}$$

X is the state covariance satisfying the linear equation

$$X = AXA^* + DWD^* \quad ; \tag{2.2}$$

Y is the output covariance and is given by

$$Y = CXC^* \ . \tag{2.3}$$

Equivalently, we may define X to be the controllability grammian associated with $(A, DW^{1/2})$; in this case there is no need to characterize $w(\cdot)$ as white noise. We shall hereafter refer to the stochastic interpretation above, even though the other valid interpretation is entirely deterministic. A deterministic interpretation may be applied to every result in the sequel.

The second class of disturbances of interest will be the class of L_2 disturbances, i.e. disturbances which satisfy

$$\sum_{k=0}^{\infty} w^*(k)w(k) < \infty \quad .$$

For a prescribed positive definite symmetric matrix W, we define the weighted L_2–norm of such a disturbance by

$$\|w(\cdot)\|_2 \triangleq \Big[\sum_{k=0}^{\infty} w^*(k)W^{-1}w(k)\Big]^{1/2} \tag{2.4}$$

and we define the L_∞–norm of the output by

$$\|y(\cdot)\|_\infty \triangleq \sup_{k \ge 0} \, [y^*(k)y(k)]^{1/2} \quad . \tag{2.5}$$

The following theorem states that $\sigma[Y]^{1/2}$ is an upper bound on the input-output gain of system (2.1) when the inputs are L_2 and the outputs are L_∞.

Theorem 1.

Consider a discrete-time system described by (2.1) *with* $x(0) = 0$ *and suppose* A *is asymptotically stable and* (A, D) *is controllable. Let*

$$Y \triangleq CXC^*$$

where X *is the unique positive definite symmetric solution of*

$$X = AXA^* + DWD^*$$

Then for any L_2 *disturbance* $w(\cdot)$,

$$\|y(\cdot)\|_\infty^2 \le \bar{\sigma}[Y]\|w(\cdot)\|_2^2 \quad . \tag{2.6}$$

Proof.

Note that $y(\cdot)$ in (2.1) and Y in (2.3) do not depend on the choice of state variables. Hence $\bar{\sigma}[Y]$ does not. This proves that if (2.6) holds for any basis, it holds for every choice of state variables. Consider now the coordinates in which $X = I$. (This is possible for every $X > 0$ by using the coordinate transformation $x = Tz$, where $TT^* = X$. Then $\lim_{k\to\infty} E[z(k)z^*(k)] = T^{-1}XT^{-*} = I$). In this case, (2.2) and (2.3) reduce to

$$I = AA^* + DWD^* \tag{2.7}$$

$$Y = CC^* \tag{2.8}$$

respectively. Utilizing (2.1), it follows that for any $k \ge 0$,

$$\|x(k+1)\|^2 = \|x(k)\|^2 + \|w(k)\|_{W^{-1}}^2 - R(x(k), w(k)) \tag{2.9}$$

where

$$R(x,w) \triangleq x^*(I - A^*A)x - 2x^*A^*Dw + w^*(W^{-1} - D^*D)w. \tag{2.10}$$

We now show that $R(x,w) \ge 0$. To this end choose $z \in \mathbb{R}^n$ and $v \in \mathbb{R}^{n_w}$ such that

$$w = WD^*z + W^{1/2}v \quad ; \quad DW^{1/2}v = 0 \ . \tag{2.11}$$

Hence, recalling (2.7),

$$\begin{aligned} x^*A^*Dw &= x^*A^*DWD^*z \\ &= x^*A^*(I - AA^*)z \\ &= x^*(I - A^*A)A^*z \ . \end{aligned} \tag{2.12}$$

Also,

$$w^*(W^{-1} - D^*D)w = z^*DW(W^{-1} - D^*D)WD^*z + \|v\|^2 \ . \tag{2.13}$$

Utilizing (2.7),

$$\begin{aligned} DW(W^{-1} - D^*D)WD^* &= DWD^* - DWD^*DWD^* \\ &= I - AA^* - (I - AA^*)(I - AA^*) \\ &= AA^* - AA^*AA^* \\ &= A(I - A^*A)A^* \ , \end{aligned}$$

i.e.,

$$DW(W^{-1} - D^*D)WD^* = A(I - A^*A)A^* \ . \tag{2.14}$$

Substituting (2.12 - 2.14) into (2.10) yields

$$\begin{aligned} R(x,\ w) &= x^*(I - A^*A)x - 2x^*(I - A^*A)A^*z + z^*A(I - A^*A)A^*z + \|v\|^2 \\ &= (x - A^*z)^*(I - A^*A)(x - A^*z) + \|v\|^2 \ . \end{aligned} \tag{2.15}$$

From (2.7), we have

$$AA^* \le I \ ;$$

hence

$$A^*A \le I$$

and

$$R(x,w) \ge 0 \ .$$

Thus, for any $k \ge 0$, it follows from (2.9) that,

$$\|x(k{+}1)\|^2 \le \|x(k)\|^2 + \|w(k)\|^2_{W^{-1}} \ .$$

Since $\|x(0)\| = 0$, we must have

$$\begin{aligned} \|x(k)\|^2 &\le \sum_{j=0}^{k-1} \|w(k)\|^2_{W^{-1}} \\ &\le \sum_{j=0}^{\infty} \|w(k)\|^2_{W^{-1}} \\ &= \|w(\cdot)\|^2_2 \ . \end{aligned} \tag{2.16}$$

It follows now that, for all $k \ge 0$,

$$\begin{aligned} \|y(k)\|^2 &= \|Cx(k)\|^2 \\ &\le \|C\|^2\|x(k)\|^2 \\ &\le \|CC^*\| \ \|w(\cdot)\|^2_2 \\ &= \bar{\sigma}[Y]\|w(\cdot)\|^2_2 \ ; \end{aligned}$$

hence

$$\|y(\cdot)\|^2_\infty \le \bar{\sigma}[Y]\|w(\cdot)\|^2_2 \qquad \square$$

Example.

Let

$$A=\begin{bmatrix}0.9671 & 0.1484\\ -0.2968 & 0.5219\end{bmatrix} \quad ; \quad D=\begin{bmatrix}0.0164\\ 0.1484\end{bmatrix} \quad ; \quad C=\begin{bmatrix}1.0000 & 0.0000\end{bmatrix}$$

and

$$\|w(\cdot)\|_2^2 = 1.0 \quad ; \quad W = 1.0 \ .$$

Using (2.2) and (2.3), we obtain

$$X=\begin{bmatrix}0.0166 & 0.0003\\ 0.0003 & 0.0321\end{bmatrix} \quad ; \quad Y = 0.0166 \ .$$

Then for any L_2 disturbance w with $\|w(\cdot)\|_2^2 \le 1.0$ we have

$$\|y(\cdot)\|_\infty^2 \le 0.0166 \ .$$

3. PARAMETER VARIATIONS AND COVARIANCES

In this section, we present some stability robustness measures for unstructured and structured parameter variations in terms of state covariances. Consider an uncertain discrete-time system described by

$$\begin{aligned} x(k+1) &= (A+\Delta A)x(k) + Dw(k)\\ y(k) &= Cx(k) \end{aligned} \tag{3.1}$$

where ΔA represents uncertainty in the system description due to parameter variations. If $w(\cdot)$ is zero mean white noise with intensity $W > 0$, the nominal state covariance is uniquely given by

$$X = AXA^* + DWD^* \ . \tag{3.2}$$

The following theorem yields two stability robustness measures for (3.1) in terms of X.

Theorem 2.

Consider a system described by (3.1) *with* A *asymptotically stable and* D *nonsingular. Suppose* X *is the solution of* (3.2) *and* $TT^* = X$. *Then* (3.1) *is asymptotically stable if*

i) $\bar{\sigma}[T^{-1}\Delta AT] < \beta$ (3.3)

or

ii) $\bar{\sigma}[\Delta A] < \beta \left[\dfrac{\underline{\sigma}[X]}{\bar{\sigma}[X]}\right]^{1/2} \le \beta$ (3.4)

where

$$\begin{aligned} \beta &\triangleq 1 - \bar{\sigma}[T^{-1}AT]\\ &= 1 - \{1 - \underline{\lambda}[DWD^*X^{-1}]\}^{1/2} \ . \end{aligned} \tag{3.5}$$

Proof:

Let $X = TT^*$ and transfer (3.1) to new coordinates $\hat{x}(k) \triangleq T^{-1}x(k)$ so that $\hat{X} = T^{-1}XT^{-*} = I$, $\hat{A} = T^{-1}AT$, $\hat{D} = T^{-1}D$, $\hat{C} = CT$, and $\Delta\hat{A} = T^{-1}\Delta AT$. Since coordinate transformations do not change stability, it is clear that (3.1) is asymptotically stable if, for

$\hat{x}(k) \neq 0$,

$$\|\hat{x}(k+1)\| = \|(\hat{A} + \Delta\hat{A})\hat{x}(k)\| < \|\hat{x}(k)\| \,. \tag{3.6}$$

Let $\|(\cdot)\| \triangleq \bar{\sigma}(\cdot)$. Then (3.6) holds if

$$\|\hat{A} + \Delta\hat{A}\| < 1 \quad . \tag{3.7}$$

Since $\|\hat{A} + \Delta\hat{A}\| \leq \|\hat{A}\| + \|\Delta\hat{A}\|$, (3.6) holds if

$$\|\Delta\hat{A}\| + \|\hat{A}\| = \|T^{-1}\Delta AT\| + \|T^{-1}AT\| < 1 \,\,. \tag{3.8}$$

Eq. (3.8) verifies (3.3).

Since,

$$\begin{aligned} \|T^{-1}\Delta AT\| &\leq \|\Delta A\| \, \|T^{-1}\| \, \|T\| \\ \|T^{-1}\| &= \bar{\sigma}[X^{-1}]^{1/2} = \underline{\sigma}[X]^{-1/2} \\ \|T\| &= \bar{\sigma}[X]^{1/2} \;\;, \end{aligned} \tag{3.9}$$

then using (3.3), we can verify (3.4)

$$\|\Delta A\| < \beta \left[\frac{\underline{\sigma}[X]}{\bar{\sigma}[X]}\right]^{1/2}$$

Note that

$$\begin{aligned} \|T^{-1}AT\|^2 &= \|T^{-1}ATT^*A^*T^{-*}\| \\ &= \|T^{-1}AXA^*T^{-*}\| \\ &= \|T^{-1}(X - DWD^*)T^{-*}\| \\ &= \|I - T^{-1}DWD^*T^{-*}\| \\ &= \bar{\lambda}[I - T^{-1}DWD^*T^{-*}] \\ &= 1 - \underline{\lambda}[T^{-1}DWD^*T^{-*}] \\ &= 1 - \underline{\lambda}[DWD^*X^{-1}] \end{aligned} \tag{3.10}$$

Using $\|T^{-1}AT\| = \{1 - \underline{\lambda}[DWD^*X^{-1}]\}^{1/2}$, then,

$$\begin{aligned} \beta &\triangleq 1 - \|T^{-1}AT\| \\ &= 1 - \{1 - \underline{\lambda}[DWD^*X^{-1}]\}^{1/2} \qquad \square \end{aligned}$$

Since $\bar{\sigma}[X^{-1}] = (\underline{\sigma}[X])^{-1}$, the right hand side of (3.4) is inversely proportional to the square root of the condition number of the state covariance matrix X, defined by Cond (X) $\triangleq \bar{\sigma}[X]\bar{\sigma}[X^{-1}]$. In addition, note that

$$1 \leq \text{Cond}\ (X) \leq \infty$$

and (3.4) can be written as

$$\bar{\sigma}[\Delta A] < \beta(\text{Cond}\ [X])^{-1/2} \ .$$

Theorem 2 gives a simple way to obtain stability robustness bounds for unstructured parameter variations. Consider now structured parameter variations defined by

$$\Delta A = \sum_{i=1}^{m} p_i \Delta A_i \tag{3.11}$$

where ΔA_i (i = 1,2,...,m) are known matrices and p_i (i = 1,2,...,m) are uncertain parameters. We have the following corollary.

Corollary of Theorem 2.

For the discrete system (3.1), *let* ΔA *represent structured parameter variations defined by* (3.11). *Assume that* A *is asymptotically stable,* D *is nonsingular, and define the state covariance to be* X *as given by* (3.2). *Then* (3.1) *is asymptotically stable if*

$$\sum_{i=1}^{m} p_i^2 < \beta^2/\bar{\sigma}^{2}[S] \tag{3.12}$$

where

$$S = T^{-1}[\Delta A_1, \Delta A_2, ..., \Delta A_m]T .$$

Proof:

With $K = [p_1 I, p_2 I, ..., p_m I]^*$,

$$\begin{aligned} \|T^{-1}\Delta AT\|^2 &= \|\sum_{i=1}^{m} p_i T^{-1}\Delta A_i T\|^2 \\ &= \|SK\|^2 \\ &\le \|S\|^2\|K\|^2 \\ &= (\sum_{i=1}^{m} p_i^2)\bar{\sigma}^{2}[S] \quad . \end{aligned} \tag{3.13}$$

Using (3.13)

$$\sum_{i=1}^{m} p_i^2 < \beta^2/\bar{\sigma}^{2}[S]$$

implies (3.3) □

The following is an example illustrating the use of Theorem 2.

Example.

Let

$$A = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad ; \quad DWD^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} .$$

Using (3.2),

$$X = \begin{bmatrix} 4/3 & 0 \\ 0 & 4/3 \end{bmatrix} .$$

Using (3.5), we get

$\beta = 0.5$

and $\beta \left[\frac{\underline{\sigma}(X)}{\bar{\sigma}(X)}\right]^{½} = 0.5$

So, if $\|\Delta A\| < 0.5$, the given system is asymptotically stable. Note that for this example, Theorem 2 gives the largest bound.

4. CONCLUSIONS

This paper shows that the following robustness properties of discrete-time systems are directly related to the system covariance matrix: robustness to uncertain disturbances, to uncertain structured parameters, to uncertain unstructured parameters, and to L_∞

performance requirements.

Future research on controller synthesis is desired to take further advantage of the explicitness of covariance controllers in terms of the state covariance.

REFERENCES

[1] C. Hsieh and R.E. Skelton, "All Covariance Controllers for Discrete Time Systems," IEEE Trans. Auto. Contr., to appear.

[2] S. R. Kolla, R. K. Yedavalli, and J. B. Farison, "State Space Model Perturbation Bounds for Robust Stability of Linear Discrete-Time Systems," Int. J. Contr., 1989, to appear.

[3] M. Corless, G. Zhu, and R. Skelton, "Improved Robustness Bounds Using Covariance Matrices," 28th IEEE Conference on Decision and Control, Tampa, Florida, 1989.

ROBUSTNESS OF DISCRETE-TIME SYSTEMS IN THE PRESENCE OF UNSTRUCTURED STOCHASTIC PERTURBATIONS

ENGIN YAZ
Electrical Engineering Dept.
University of Arkansas
Fayetteville, AR 72701

INTRODUCTION

This work contains derivation of novel measures for stochastic stability robustness of discrete-time control systems. Most robust control methods are characterized by treating perturbations as deterministic but unknown quantities with available upper bounds on their magnitudes. Descriptions of frequency domain and time domain techniques can be found in [1]. In this work, we pursue the time-domain approach. By using positive super-martingale convergence results, we will demonstrate the ability of nominally exponentially stable systems to maintain stability in the presence of unstructured stochastic (linear and nonlinear) perturbations. This work is an extension of the stochastic results in [2] to perturbation models with no prespecified structure and the results of [3] to the stochastic perturbations.

MAIN RESULTS

We consider the following nominal system

$$z_{k+1} = Az_k \tag{1}$$

where $A \epsilon R^{nxm}$ has all its eigenvalues inside a disc of radius $(1 + \alpha)^{-1/2}$, $\alpha > 0$. This stability margin will be used in the derivation of measures of stochastic robustness. First, we assume that the system is affected by stochastic perturbations (with no structural knowledge), so that we have

$$x_{k+1} = Ax_k + f_k(x_k, \omega) \tag{2}$$

The only information available about the perturbation term f_k is a bound on its second moment as given below:

$$E\{||f_k(x_k,\omega)||^2\} \leq \mu^2 \; E\{||x_k||^2\}, \; 0 < \mu < \infty \tag{3}$$

This model is effective for cone bounded nonlinearities (saturation, deadzone, piecewise-linear, etc.) multiplied by white noise. Hence, the problem becomes that of determining the largest bound μ which can be tolerated by the system without causing instability. Obviously, this is a function of the stability degree of the original system. Because of the stability degree, there exists a unique solution $P = P^T > 0$ to

$$(1 + \alpha)A^T PA - P = -Q \tag{4}$$

for any matrix $Q = Q^T > 0$. We present our stochastic stability-robustness results below:

Theorem 1. The perturbed system (2) is mean square (m.s.) and almost surely (a.s.) exponentially stable if

$$\mu < \mu_1 = |\alpha\underline{\lambda}(Q)(1+\alpha)^{-1} \; \bar{\lambda}^{-1} \; (P)]^{1/2} \tag{5}$$

where P is the solution to (4), $\underline{\lambda}(.)$ $(\lambda(.))$ denote minimum (maximum) eigenvalue of a symmetric matrix.

Remark. Since it is shown in [3] that the bound μ_1 is maximum when $Q = I$ is chosen, from now on, we will assume so in our discussion.

Theorem 2: The perturbed system (2) is m.s. and a.s. exponentially stable if

$$\mu < \mu_2 = -\|A\|_s + \{\|A\|_s^2 + [\alpha\, \underline{\lambda}(P) + 1]\ (1+\alpha)^{-1}\ \bar{\lambda}^{-1}(P)\}^{1/2} \qquad (6)$$

where $\|A\|_s$ denotes the spectral norm of matrix A.

Theorem 3. The perturbed system (2) is m.s. and a.s. exponentially stable if

$$\mu < \mu_3 = [0.5\ \alpha\beta^{-2}(1+\alpha)^{-1}]^{1/2} \qquad (7)$$

where $\beta > 0$ is such that $\|A^k\|_s \leq \beta(1 + \alpha)^{-k/2}$.

Corollary 1. Let A be diagonalizable and

$$\mu < \mu_4 = [0.5\ \alpha\ \gamma^{-2}\ (1+\alpha)^{-1}]^{1/2} \qquad (8)$$

hold, where $\gamma = \|T\|_s\ \|T^{-1}\|_s$ with T being the similarity transformation that diagonalizes A, then the perturbed system (2) is m.s. and a.s. exponentially stable. If A is a normal matrix, then $\gamma = 1$.

We will now be interested in the linear perturbations of the following form

$$f_k(x_k,\ \omega) = A_k(\omega)x_k \qquad (9)$$

where $A_k(\omega)$ is a matrix whose elements are second order white noise sequences. Since, this constitutes a special case of our more general perturbation model (2)-(3), we will use our previous results to find bounds on the second moments of $A_k(\omega)$.

Corollary 2. System (2) with linear perturbation (9) is m.s. and a.s. exponentially stable if any one of the following conditions holds:

i) $\sup_k \bar{\lambda}(E\{A_k^T A_k\}) < \max\{\mu_1^2,\ \mu_2^2,\ \mu_3^2,\ \mu_4^2\} \triangleq m$

ii) $\sup_k \mathrm{tr}[E\{A_k^T A_k\}] < m$

iii) $\sup_k \max_{i,j} [[E\{A_k^T A_k\}]_{ij}] < m/n^2$

REFERENCES

[1] P. Dorato, IEEE Contr. Syst. Mag., vol. 7, pp. 44-47, 1987.
[2] E. Yaz, IEEE Trans. Automat. Contr., vol. AC-33, pp. 952-955, 1988.
[3] E. Yaz and X. Niu, Proc. ACC, pp. 1610-1614, 1989.

ROBUST LINEAR STATE FEEDBACK STABILIZATION OF CONTINUOUS TIME UNCERTAIN DYNAMIC SYSTEMS

ZHIHUA QU and JOHN DORSEY
School of Electrical Engineering,
Georgia Institute of Technology, Atlanta, GA30332, USA
This work is supported in part by K.C.WONG Education Foundation Ltd.,
Hong Kong

ABSTRACT

Stabilization of continuous time systems with both state and input dependent uncertainties using a purely linear state feedback control law is explored. Sufficient conditions for stability are given in term of bounds for the time-varying coefficients of state and input dependent uncertainties. The effect of linear feedback gain on stability is discussed, and optimal and suboptimal design of a robust feedback control law to maximize the stable region is investigated.

1. Introduction

The problem of determining a feedback control law which stabilizes a linear system containing deterministic time-varying uncertainty has received considerable attention. Nonlinear controllers which asymptotically or uniformly ultimately stabilize an uncertain continuous time system are given in [1,2,4,5,10]. Thorp and Barmish[11] proposed a linear feedback control law to guarantee stability of an uncertain linear system under the assumption that the input matrix B has maximal rank. A requirement common to most of the work done to date is that the system must satisfy the so-called "matching conditions". There have been some attempts to loosen this serious limitation[8], but the problem is far from solved. An alternative method, called the Riccati approach[3], does not require matching conditions, but instead uncertainties must, by assumption, be structured and of rank-1 type.

In this paper we consider the problem of robust stabilization of a class of continuous time uncertain dynamical systems where the "nominal system" is linear and the uncertainties are of deterministic nature and depend on both system states and control inputs. The only additional assumption made is that the uncertainties are cone-bounded. Sufficient conditions have been derived for stability, or more precisely, for ultimate boundedness. In this general case, it happens that, increasing the feedback gain, decreases the upper and lower bounds of the stable region because one of the uncertainties is driven by the control input. Therefore an optimal robust control design is considered to maximize the stable region. In the case that the matching conditions are satisfied, a new linear feedback control law that always stabilizes the system is proposed that avoids any requirement on the rank of the input matrix, B.

2. Problem Formulation

We shall consider linear continuous-time dynamical systems modeled by the following differential equation:

$$\dot{x}_t = (A + \Delta A(r_t))x_t + (B + \Delta B(\nu_t))u_t + E(v_t) \tag{1}$$

where $x_t \in \mathbf{R}^n$, $u_t \in \mathbf{R}^m$, A and B are constant matrices of appropriate dimensions, $\Delta A(\cdot) : \mathbf{R}^{l_1} \to \mathbf{R}^{n\times n}$, $\Delta B(\cdot) : \mathbf{R}^{l_2} \to \mathbf{R}^{n\times m}$, and $E(\cdot) : \mathbf{R}^{l_3} \to \mathbf{R}^{n\times 1}$ are continuous time-invariant functions.

The uncertainties are determined by the variables $r(\cdot)$, $\nu(\cdot)$ and $v(\cdot)$, whose behavior we do not know at any time $t \in \mathbf{R}^+$, the set of non-negative real numbers. It is assumed, however, that they are constrained to known closed and bounded sets, namely,

$$r_t \in S_1 \subset \mathbf{R}^{l_1}\ , \nu_t \in S_2 \subset \mathbf{R}^{l_2}\ , \text{and } v_t \in S_3 \subset \mathbf{R}^{l_3}.$$

Therefore, equation (1) can be rewriten in the form

$$\dot{x}_t = Ax_t + Bu_t + e(t, x_t, u_t) \tag{2}$$

where $e(t, x_t, u_t) = \Delta A(r_t)x_t + \Delta B(\nu_t)u_t + E(v_t)$.

It then follows that

$$\begin{aligned} \|e(t, x_t, u_t)\| &= \|\Delta A(r_t)x_t + \Delta B(\nu_t)u_t + E(v_t)\| \\ &\le \max_{r_t\in S_1} \|\Delta A(r_t)\| \cdot \|x_t\| + \max_{\nu_t\in S_2} \|\Delta B(\nu_t)\| \cdot \|u_t\| + \max_{v_t\in S_3} \|E(v_t)\|. \end{aligned}$$

To simplify the notation somewhat, let

$$\xi(t, x_t, u_t) = \xi_0 + \xi_1\|x_t\| + \xi_2\|u_t\| \tag{3}$$

where $\xi_0 = \max_{v_t\in S_3} \|E(v_t)\|$, $\xi_1 = \max_{r_t\in S_1} \|\Delta A(r_t)\|$, and $\xi_2 = \max_{\nu_t\in S_2} \|\Delta B(\nu_t)\|$.

Then,

$$\|e(t, x_t, u_t)\| \le \xi(t, x_t, u_t). \tag{4}$$

where $\|\cdot\|$ is the Euclidean norm.

In the subsequent discussion we use the induced matrix norm $\|M\| = [\lambda_{max}(M^T M)]^{\frac{1}{2}}$, where $\lambda_{max}(\cdot)$ denotes the operation of taking the largest eigenvalue.

We make the following assumptions for systems (1):

(i) The nominal system $\dot{x}_t = Ax_t + Bu_t$ is stabilizable.

(ii) The uncertainty $e(t, x_t, u_t)$ is cone bounded as defined in (4).

3. Linear State Feedback Control Law

By assumption (i) there exists a constant feedback gain matrix $K \in \mathbf{R}^{m\times n}$ such that $A - BK$ is asymptotically stable with prescribed pole placement. That is, letting $u_t = -Kx_t$, the system (1) becomes

$$\dot{x}_t = \overline{A}x_t + e(t, x_t, Kx_t) \tag{5}$$

where $\overline{A} = A - BK$.

Then there is a real symmetric positive definite(r.s.p.d.) matrix Q such that a r.s.p.d. matrix P uniquely solves the Lyapunov matrix equation

$$\overline{A}^T P + P\overline{A} = -Q. \tag{6}$$

Let

$$V(x_t) = x_t^T P x_t = \|x_t\|_P^2 \tag{7}$$

be a Lyapunov function for system (5). It then follows that

$$\begin{aligned} \dot{V}(x_t) &= -x_t^T Q x_t + 2x_t^T P e(t, x_t, K x_t) \\ &\leq -\lambda_{min}(Q)\|x_t\|^2 + 2\|x_t\| \cdot \|P\| \cdot \|e(t, x_t, u_t)\| \\ &\leq -\alpha_1\|x_t\|^2 + 2\alpha_2\|x_t\| + \alpha_3. \end{aligned}$$

where

$$\begin{aligned} \alpha_1 &= \lambda_{min}(Q) - 2\lambda_{max}(P) \cdot (\xi_1 + \|K\|\xi_2) \\ \alpha_2 &= \xi_0\lambda_{max}(P) \quad \alpha_3 = 0 \end{aligned}$$

The following theorem develops the stability properties of the uncertain continuous time system (1) under purely linear state feedback control for $\alpha_1 > 0$. The nonnegative constant α_3 is introduced in the above inequality of $\dot{V}$ since it will be required in the subsequent analysis. The proof is omitted for briefness since it is completely analogous to those given in [4,6], for the case where the matching conditions are assumed.

Proposition: Let $f(y) = -\alpha_1 y^2 + 2\alpha_2 y + \alpha_3$ be a quadratic polynomial with $\alpha_1 > 0$ and $\alpha_2, \alpha_3 \geq 0$. For $y \geq 0$, $f(y) < 0$ if $y > \eta$ where

$$\eta = \frac{\sqrt{\alpha_2^2 + \alpha_1\alpha_3} + \alpha_2}{\alpha_1}$$

Theorem 1: System (5) with cone bounded disturbance $e(t, x_t)$ and with $\alpha_1 > 0$ has the following properties:
(1) Uniformly bounded, namely, if $x(\cdot) : [t_0\,, t_1] \to \mathbf{R}^n$ is a solution to (5) with $x(t_0) = x_0$, then

$$\|x_0\| \leq s \Rightarrow \|x_t\| \leq d(s), \ \forall t \in [t_0\,, t_1]$$

where

$$d(s) = \begin{cases} \delta\, s & \text{if } s > \eta \\ \delta\, \eta & \text{if } s \leq \eta \end{cases},$$

with η defined as in the above Proposition, and

$$\delta = \left(\frac{\lambda_{max}(P)}{\lambda_{min}(P)}\right)^{\frac{1}{2}}.$$

(2) Uniformly ultimately bounded, namely, if $x(\cdot) : [t_0, \infty) \to \mathbf{R}^n$ is a solution to (5) with $x(t_0) = x_0$ and with $\|x_0\| \leq s$, then given $d_1 > \delta\eta$, $\|x_t\| \leq d_1$, $\forall t \in [t_0 + T(d_1, s), \infty)$, where

$$\begin{aligned} T(d_1, s) &= \begin{cases} 0 & \text{if } s \leq \eta_1 \\ \frac{\lambda_{max}(P)s^2 - \lambda_{min}(P)\eta_1^2}{w\eta_1^2} & \text{if } s > \eta_1 \end{cases}, \\ \eta_1 &= \frac{d_1}{\delta}, \end{aligned}$$

and

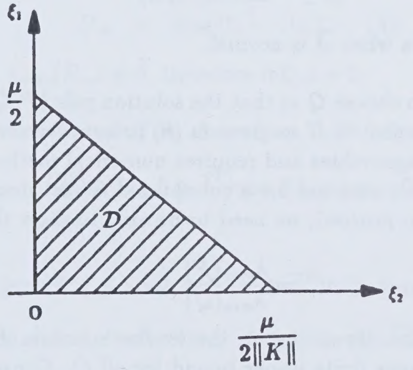
$$w = \frac{\alpha_1\|\eta_1\|^2 - 2\alpha_2\|\eta_1\| - \alpha_3}{\|\eta_1\|^2}$$

which is the negative of the upper bound of $\dot{V}(x_t)$,
(3) If $\alpha_1 > 0$ and $\alpha_2 = \alpha_3 = 0$(i.e. $\xi_0 = 0$ in this case), then the origin is uniformly asymptotically stable in the large.

4. Evaluation of Stability Region for Uncertainties

We have shown that the closed-loop system (5) is stable if $\alpha_1 > 0$. The next step is to determine the region in the uncertainty space defined by ξ_1 and ξ_2, in which the linear feedback control law can stabilize the system.

If A, B and K are given, then α_1 is a function of ξ_1 and ξ_2. Therefore it follows that the linear state feedback control law can stabilize the system provided that uncertainties lie within the shaded area of the figure, shown below.



where

$$\mu \triangleq \frac{\lambda_{min}(Q)}{\lambda_{max}(P)}.$$

If there are extra degrees of freedom in the controller, additional criteria other than prescribed pole placement can be achieved.

The robust design problem will be the following. Given A, B and a prescribed pole placement, namely, that $\lambda_i(\overline{A}) = \lambda_1, \cdots, \lambda_n$, find K that will maximize the area $\mathcal{D}$ where

$$\mathcal{D} = \frac{1}{8}\frac{\mu^2}{\|K\|}. \tag{8}$$

The ratio μ plays an important role in the stablity analysis. It follows from (5) and (6) that μ depends on the choice of the matrix Q in (6) and the choice of K in $\mathbf{K}$, where $\mathbf{K}$ is a subset of $\mathbf{R}^{m\times n}$ such that, given A and B, $\overline{A}$ has the same prescribed eigenvalues for any $K \in \mathbf{K}$. The following lemma shows how the choice of Q affects μ and clarifies relationship between the upper bound of μ and the dominant eigenvalue(s) of $\overline{A}$.

Lemma 1:[12] Given an r.s.p.d. matrix Q, let P be the unique r.s.p.d. solution of equation (6), and let P_1 be the unique r.s.p.d. solution of the Lyapunov equation

$$\overline{A}^T P_1 + P_1\overline{A} = -Q_1 = -qQ$$

where $q > 0$ is a scalar. Then

$$\mu = \frac{\lambda_{min}(Q)}{\lambda_{max}(P)} = \frac{\lambda_{min}(Q_1)}{\lambda_{max}(P_1)} = \mu_1.$$

Lemma 2:[12] The ratio μ in Lemma 1 is maximum when $Q = I_{n\times n}$. If $\overline{A}$ is asymptotically stable, then this maximum of μ has its upper bound as

$$\mu \leq -\max_i \mathrm{Re}\lambda_i(\overline{A}). \tag{9}$$

Moreover, equality holds when $\overline{A}$ is normal.

Lemma 2 tells us how to choose Q so that the solution pair (P, Q) makes $\mathcal{D}$ maximal. But maximizing $\mathcal{D}$ with respect to K as given in (8) presents some difficulties since $\mathcal{D}$ is a function depending on eigenvalues and requires numerical methods. However, $\mathcal{D}$ still has appeal as a performance measure for a suboptimal robust design, if the expression for $\mathcal{D}$ can be simplified. To proceed, we need to be able to show that

$$\mu^{-1} = \frac{\lambda_{max}(P)}{\lambda_{min}(Q)} \leq c_1 \tag{10}$$

where c_1 is a finite constant depending on the feedback matrix K. The next Lemma shows that μ^{-1} does not have finite upper bound for all Q. Consequently, to obtain a finite c_1 we choose the Q which is best suited to for our purposes.

Lemma 3: The infimum of μ in Lemma 1 is zero.
Proof: In equation (6) let Q be any arbitrary r.s.p.d. matrix and let $q = \frac{1}{\lambda_{max}(Q)}$ in Lemma 1. Then $\lambda_{max}(Q_1) = 1$ and $\mu = \mu_1$. The solution pair (P_2, Q_1^2) of the corresponding Lyapunov equation can be written as

$$P_2 = \int_0^\infty e^{\overline{A}^T t} Q_1^2 e^{\overline{A}t} dt$$

with $\mu_2 = \frac{\lambda_{min}^2(Q_1)}{\lambda_{max}(P_2)}$. Then

$$\begin{aligned} P_2 - \lambda_{min}(Q_1)P &= \int_0^\infty e^{\overline{A}^T t}(Q_1^2 - \lambda_{min}(Q_1)Q_1)e^{\overline{A}t} dt \\ &= \int_0^\infty e^{\overline{A}^T t} U^T (D^2 - \lambda_{min}(Q_1)D)U e^{\overline{A}t} dt \\ &\geq 0 \end{aligned}$$

where $Q_1 = U^T D U$, D is diagonal and U is unitary.

We then have $\lambda_{max}(P_2) \geq \lambda_{min}(Q_1)\lambda_{max}(P)$ where the equality holds if and only if $Q_1 = cI$ where c is an any real constant. To find minimum(or infimum) of μ, we can, by Lemma 2, wolog assume that $Q_1 \neq cI$(i.e. $\lambda_{min}(Q_1) < 1$). Hence

$$\mu_2 = \frac{\lambda_{min}^2(Q_1)}{\lambda_{max}(P_2)} < \frac{\lambda_{min}(Q_1)}{\lambda_{max}(P)} = \mu$$

Now let (P_n, Q_1^n) $(n = 1, 2, \cdot)$ be the solution pairs of their corresponding Lyapunov equations, it follows by induction that

$$\mu_n = \frac{\lambda_{min}^n(Q_1)}{\lambda_{max}(P_n)} < \mu$$

Since $\lambda_{min}(Q_1) < 1$, we have that $\lim_{n\to\infty} \lambda_{min}^n(Q_1) = 0$ and that $\lim_{n\to\infty} \lambda_{max}(P_n) = \lambda_{max}(P_\infty)$ where

$$\begin{aligned} P_\infty &= \int_0^\infty e^{\overline{A}^T t} D_\infty e^{\overline{A}t} dt \\ D_\infty &= \text{diag}\{0, \cdots, 0, 1, \cdots, 1\} \end{aligned}$$

It is obvious that $\lambda_{max}(P_\infty) \neq 0$, therefore $\inf_Q \mu = 0$. □

we next examine the effect of K on the ratio μ^{-1}. It follows from (6) that

$$(\overline{A}^T \otimes I + I \otimes \overline{A}^T)\text{vec}(P) = -\text{vec}(Q).$$

Thus

$$\|\text{vec}(P)\| \leq \|(\overline{A}^T \otimes I + I \otimes \overline{A}^T)^{-1}\| \cdot \|\text{vec}(Q)\|.$$

That is, since $\|\text{vec}(\cdot)\| = \|\cdot\|_F$ and since $\|\cdot\| \leq \|\cdot\|_F \leq \sqrt{n}\|\cdot\|$,

$$\begin{aligned} \lambda_{max}(P) = \|P\| \leq \|P\|_F &\leq \frac{1}{\underline{\sigma}(\overline{A}^T \otimes I + I \otimes \overline{A}^T)}\|Q\|_F \\ &\leq \frac{n}{\underline{\sigma}(\overline{A}^T \otimes I + I \otimes \overline{A}^T)}\|Q\| \end{aligned}$$

where $\otimes$ is Kronecker product, and $\|Y\|_F$ is Frobenius norm of the matrix augument Y.

Thus,

$$\frac{\lambda_{max}(P)}{\lambda_{min}(Q)} \leq \frac{n}{\underline{\sigma}(\overline{A}^T \otimes I + I \otimes \overline{A}^T)} \cdot \frac{\lambda_{max}(Q)}{\lambda_{min}(Q)}. \tag{11}$$

For any matrix $W \in \mathbf{R}^{n \times n}$, it can be easily shown that

$$\prod_{i=1}^{n} \sigma_i^2(W) = \prod_{j=1}^{n} |\lambda_j(W)|^2$$

Letting $W = \overline{A} \otimes I + I \otimes \overline{A}^T$, the right hand side of above equation is fixed and finite by the prescribed pole placement of $\overline{A}$, since $\lambda_k(\overline{A}^T \otimes I + I \otimes \overline{A}^T) = \lambda_i(\overline{A}) + \lambda_j(\overline{A})$, $i, j = 1, \cdots, n$, $k = 1, \cdots, n^2$. On the other hand, we have

$$\begin{aligned}
\overline{\sigma}(\overline{A}^T \otimes I + I \otimes \overline{A}^T) &\leq \overline{\sigma}(\overline{A}^T \otimes I) + \overline{\sigma}(I \otimes \overline{A}^T) \\
&= 2\overline{\sigma}(\overline{A}^T) \\
&= 2\|A - BK\| \\
&\leq 2(\|A\| + \|B\|\|K\|)
\end{aligned}$$

We then have

$$c_1 = n \frac{2^{n^2-1}(\|A\| + \|B\|\|K\|)^{n^2-1}}{\prod_{i,j=1}^{n} |\lambda_i(\overline{A}) + \lambda_j(\overline{A})|} \cdot \frac{\lambda_{max}(Q)}{\lambda_{min}(Q)}. \tag{12}$$

Using (8), (9), (10) and (12), we can bound the stability region $\mathcal{D}$ in the uncertainty space defined by ξ_1 and ξ_2 as follows.

$$\frac{1}{8c_1^2\|K\|} \leq \mathcal{D} \leq \frac{-\max_i \mathrm{Re}\lambda_i(\overline{A})}{8\|K\|}. \tag{13}$$

If $Q = I$ in (12), then

$$c_1 = n \frac{2^{n^2-1}(\|A\| + \|B\|\|K\|)^{n^2-1}}{\prod_{i,j=1}^{n} |\lambda_i(\overline{A}) + \lambda_j(\overline{A})|},$$

which maximizes μ and hence the lower bound of $\mathcal{D}$ (μ still depends on K).

In view of the above result, the sub-optimal robust design procedure will be to simply solve the optimization problem of minimizing $\|K\|$ with the n constraints imposed by the prescribed eigenvalues of $\overline{A}$. Equation (13) shows that this procedure does not depend on the choice of Lyapunov function, is easy to compute, and maximizes the lower and upper bounds of the stability region $\mathcal{D}$ simultaneously. As shown in the above discussion, the smaller $\|K\|$ the smaller c_1, and hence the better the lower bound. Equation (13) also shows that the stable region gets smaller as the feedback gain K gets larger which implies that it is impossible, in general, to stabilize a system by simply increasing the control gain.

Another important ratio is δ^2 defined in Theorem 1. It plays the role of "rate of convergence" or "rate of attraction", which depends on the choice of Q. If $\overline{A}$ is asymptotically stable and if $\lambda_{max}\{(\overline{A} + \overline{A}^T)Q^{-1}\} < 0$, we can use the results in [13] to obtain

$$1 \le \frac{\lambda_{max}(P)}{\lambda_{min}(P)} \le \frac{\lambda_{min}\{(\overline{A}+\overline{A}^T)Q^{-1}\}}{\lambda_{max}\{(\overline{A}+\overline{A}^T)Q^{-1}\}}.$$

If $\lambda_{max}\{(\overline{A}+\overline{A}^T)Q^{-1}\} \not< 0$, it follows from (12) and the result in [14] that

$$1 \le \frac{\lambda_{max}(P)}{\lambda_{min}(P)} \le \frac{2n\overline{\sigma}(\overline{A})}{\underline{\sigma}(\overline{A}^T \otimes I + I \otimes \overline{A}^T)} \cdot \frac{\lambda_{max}(Q)}{\lambda_{min}(Q)}.$$

Again $Q = I$ will give the best upper bound for δ^2. But, in general, $Q = I$ does not mean that the attraction rate becomes better, that is, δ^2 becomes smaller. If $\overline{A}$ is normal, both the best value of μ and the best rate attraction can be acheived at same time since one can easily check that, if $\overline{A} = U^H \Gamma U$ is the Schur decomposition of $\overline{A}$, then $[P, Q] = [I, U^H \mathrm{Re}(\Gamma) U]$ is a solution of Lyapunov equation (6) which minimizes δ and yields the same μ obtained by choosing $Q = I$.

5. Design Example

Consider the system:

$$A = \begin{bmatrix} -5 & 1 & 0 \\ -2 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

which is open-loop unstable with poles -4.494, -1.11, and 1.604. It is required to design, by partial state feedback, a robust closed-loop system with eigenvalues -4, $-2 \pm 2j$. Consider the partial state feedback controller:

$$K = \begin{bmatrix} 0 & K_{12} & K_{13} \\ 0 & K_{22} & K_{23} \end{bmatrix}$$

It is easy to show that the poles will be as prescribed if

$$\begin{cases} K_{22} + K_{13} = 4 \\ K_{23} + 2K_{13} = 0 \\ K_{22}(K_{13} - 1) - (K_{12} - 1)(K_{23} - 1) = 7 \end{cases}.$$

The sub-optimal feedback design strategyis to minimize the " magnitude" of the feedback gain:

$$\|K\|^2 = \frac{K_{12}^2 + K_{13}^2 + K_{22}^2 + K_{23}^2 + \sqrt{[(K_{22}+K_{13})^2 + (K_{12}-K_{23})^2][(K_{22}-K_{13})^2 + (K_{12}+K_{23})^2]}}{2},$$

for which the solution is

$$K_{12} = 2.481; \quad K_{13} = 1.465; \quad K_{22} = 2.535; \quad K_{23} = -2.930$$

with $\|K\| = 3.945$.

It is clear from the example that it is not very easy to minimize even the norm of the feedback gain K by analytic methods. The maximiztion of $\mathcal{D}$ would be even more involved.

Conclusion

Sufficient conditions of stability are derived for a general continuous system with both state and input dependent uncertainties under purely linear state feedback control law. This result is obtained without the assumption of matching conditions. A geometric interpretation is used to show that the feedback gain to stabilize the system should be kept as small as possible since there is an uncertainty coupled to the control input. Optimal and suboptimal design of the feedback gain are considered. In the case that the matching conditions are satisfied, a new linear control law that always stabilizes the system is proposed.

References

[1] G.Leitmann, "On the efficacy of nonlinear control in uncertain linear systems," *Journal of Dynamic Systems, Measurement, and Control*, Vol.102, pp.95-102, June, 1981.

[2] G.Leitmann, "Guaranteed asymptotic stability for some linear systems with bounded uncertainties," *Journal of Dynamic Systems, Measurement, and Control*, Vol.101, pp.212-216, Sept., 1979.

[3] H.L.Stalford, "Robust control of uncertain systems in the absence of matching conditions: scalar input," *Proceedings of the 26th CDC*, pp.1298-1307, 1987

[4] M.J.Corless and G.Leitmann, "Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems," *IEEE Trans. Automat. Contr.*, Vol.26, pp.1139-1144, 1981

[5] M.J.Corless and G.Leitmann, "Adaptive control for uncertain dynamical systems," in *Dynamical Systems and Microphysics: Control Theory and Mechanics*, Ed. by A.Blaquiere and G.Leitmann, Academic Press Inc., 1984.

[6] M.E.Magana and S.H.Zak, "Robust state feedback stabilization of discrete-time uncertain dynamical systems," *IEEE Trans. Automat. Contr.*, Vol.33, No.9, pp.887-891, 1988

[7] G.W.Stewart, *Introduction to Matrix Computations*, New York: Academic 1973

[8] C.V.Hollot and M.Arabacioglu, "l^{th}-step Lyapunov min-max controllers: Stabilizing discrete-time systems under real parameter variations," *Proceeding of ACC*, pp.496-501, 1987

[9] W.E.Schmittendorf, "Designing stabilizing controller for uncertain systems using the Riccati equation approach," *Proceeding of ACC*, pp.502-505, 1987

[10] S.Gutman, "Uncertain dynamical systems — A Lyapunov min-max approach," *IEEE Trans. Automat. Contr.*, Vol.24, No.3, pp.437-443, 1979

[11] J.S.Thorp and B.R.Barmish, "On guaranteed stability of uncertain linear systems via linear control," *J. of Optimiz. Theory and Applic.*, Vol.35, No.4, pp.559-579,1981

[12] R.V.Patel and M.Toda, "Quantitative measures of robustness for multivariable systems," *Proc. JACC*, Vol.2, TP8-A, 1980

[13] K.Yasuda and K.Hirai, "Upper and lower bounds on the solution of algebraic Riccati equation" *IEEE Trans. Automat. Contr.*, Vol.24, No.3, pp.486-487, 1979

[14] E.Y.Shapiro, "On the Lyapunov matrix equation" *IEEE Trans. Automat. Contr.*, Vol.19, pp.549-551, 1974

DISTRIBUTED SENSING FOR ROBUST CONTROL OF FLEXIBLE STRUCTURES

by

Michael F. Barsky and Douglas K. Lindner

Fiber & Electro-Optics Research Center
Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

Abstract

A control system based on a distributed sensor is proposed for the control of flexible structures. It is shown that if the distributed sensor is combined with a spatial processor, then an estimator can be designed to remove uncontrolled dynamics. The resulting control system is effectively decoupled from the uncontrolled dynamics. That is, the effects of observation spillover are eliminated. An example is given to illustrate the results, and to contrast the distributed sensor based design against a similar design based on point sensing.

I. Introduction

In this paper a control system utilizing distributed sensing technology is analyzed. The distributed sensor output is processed both spatially and temporally to generate a scalar control signal for a point actuator. The spatial processing capability removes uncontrolled dynamics from the sensor output. As a result, the temporal processoring can effectively implement a finite dimensional state feedback control law. It is shown that the resulting system is removes the observation spillover associated with the uncontrolled dynamics.

Recently, an all optical control system was presented to process images captured by a real-time holographic sensor [1]. This control system can be used for controlling the vibrations in a flexible structure such as the cantilevered beam shown in Figure 1, where the deflection of the beam is denoted as $y(x;t)$. The output of the real-time holographic sensor is $y(x;k)$, an image of the flexible structure at time k where the intensity of the image corresponds to the deflection of the structure. The control system operates on the image from the sensor to produce a single point intensity which is converted to an electrical signal to drive a single actuator. The system architecture developed in [1] is shown in Figure 2. All of the integrations and summations in Figure 2 are performed optically on spatially continuous images. This system is capable of implementing infinite-dimensional LQG control laws [2-4]. A simplified version of this system is analyzed here.

To simplify the analysis, an ideal distributed sensor is analyzed in continuous time without specific reference to the sensing and processing hardware. That is, the

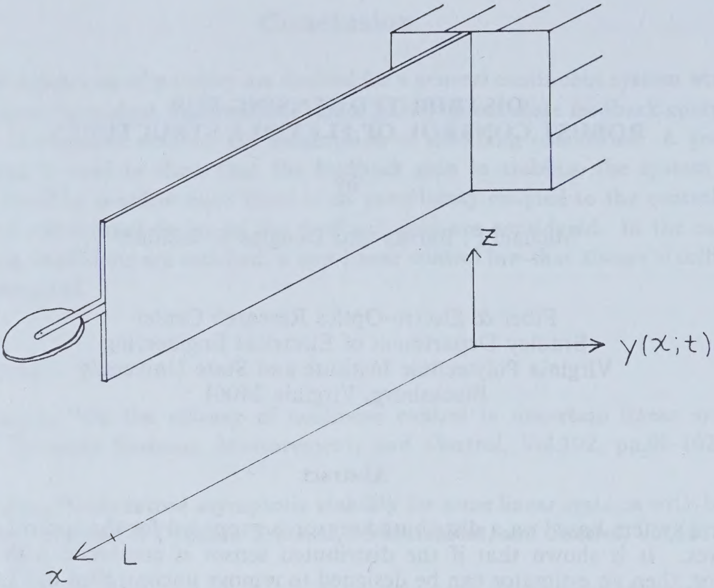


Figure 1: Cantilevered beam with tip actiator.

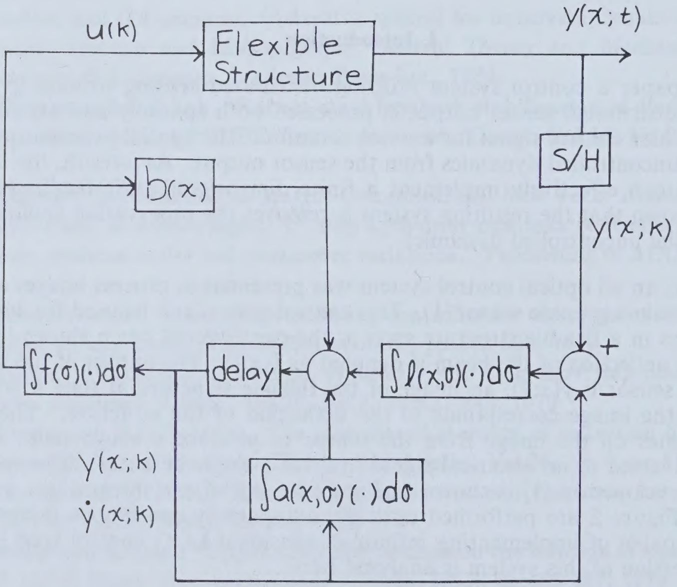

u(k)
Flexible Structure
y(x;t)
S/H
b(x)
y(x;k)
∫f(σ)(·)dσ
delay
∫l(x,σ)(·)dσ
+
−
y(x;k)
ẏ(x;k)
∫a(x,σ)(·)dσ

Figure 2: A concept for an all-optical control system to implement an infinite-dimensional LQG control law.

sensor output y is assumed to be $y(x;t)$, where the sensor biases and dynamics are neglected. Similarly, the specific spatial processing hardware will not be addressed here. It is understood that the actual sensing hardware will be holographic, and the processing hardware will be optical. Suitable optical processing hardware will be presented in [5].

In the next section we present a model for a cantilevered beam, show how the LQG design method may be applied, and analyze the closed-loop system to demonstrate the unique benefits afforded by distributed sensing and processing. In the third section an example is presented for vibration control of a cantilevered beam illustrating the degree of decoupling of the uncontrolled dynamics in distributed processing over point sensing. In the last section we summarize our results and describe areas of further study.

II. Modelling, Design, and Analysis

In this section we consider the modelling and the control system design for a cantilevered beam with a tip-force actuator. We assume that the distributed measurement can be composed from the weighted sum of N orthonormal basis functions called *modes* or *mode shapes*, where $N = N_1 + N_2$. The set of N_1 modes are used in the control system design, and are called the *controlled* modes. The remaining N_2 are not used in the design, and are refered to as *uncontrolled* modes. The number of uncontrolled modes may be taken arbitrarily large. Since the uncontrolled modes are only used to demonstrate that the closed-loop system is stable, the approximation to finite N_2 is not critical. The measurement is given by

$$y(x;t) = \sum_{i=1}^{N} p_i(x)\xi_i(t)$$

where $p_i(x)$ are the mode shapes. The time varying functions $\xi_i(t)$ are the *modal amplitudes* . The dynamics of the beam are described in state space in terms of $\xi_i(t)$. Since the beam is second order in time, the state vector consists of the modal amplitudes and their first time derivatives. Let

$$\eta_1(t) = [\xi_1(t)\ \xi_2(t) \dots \xi_{N_1}(t)\ \dot{\xi}_1(t)\ \dot{\xi}_2(t) \dots \dot{\xi}_{N_1}(t)\ ]^T$$

$$\eta_2(t) = [\xi_{N_1+1}(t)\ \xi_{N_1+2}(t) \dots \xi_{N_2}(t)\ \dot{\xi}_{N_1+1}(t)\ \dot{\xi}_{N_1+2}(t) \dots \dot{\xi}_{N_2}(t)\ ]^T$$

and

$$\eta(t) = [\eta_1^T(t)\ \eta_2^T(t)]^T.$$

Then

$$\dot{\eta}(t) = A\eta(t) + Bu(t)$$

where

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}, \text{ and } \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}.$$

The matrix A may be found from the partial differential equation for the beam in the manner described in [6]. If the beam is undamped, the eigenvalues of A are located along the imaginary axis at the natural frequencies. If damping exists, the eigenvalues of A are located to the left of their undamped positions. The vector B depends on the location and type of the actuator. The vector function $C_1^T(x)$ is given by

$$y(x;t) = [C_1(x) \;\; C_2(x)\,]\eta(t).$$

Hence,

$$C_1(x) = [p_1(x) \;\; p_2(x) \ldots p_{N_1}(x)\, 0 \ldots 0] \tag{1}$$

is $(1 \times 2N_1)$. Similarly, $C_2(x)$ is $(1 \times 2N_2)$. The zeros in $C_1(x)$ and $C_2(x)$ indicate that only the displacement is sensed, the velocity is not sensed.

The triplet (A_1, B_1, C_1) is used to generate the LQG control system. The control is given by

$$u(k) = F\hat{\eta}_1(t), \tag{2}$$

where

$$\hat{\eta}_1(t) = A_1\hat{\eta}_1(t) + B_1u(t) + \int_0^L K(x)(\,y(x;t) - C_1(x)\hat{\eta}_1(t)\,)\,dx. \tag{3}$$

Equations (2) and (3) define the usual state feedback plus observer control system except that the observer uses the distributed measurement, instead of the usual discrete measurement. The control vector F can be designed using any finite dimensional control method such as LQR or pole placement. The design of the observer can also be accomplished using standard finite dimensional control methods, but the application is less straightforward. We discuss the observer design in detail.

The design of the observer begins by spatially sampling the measurement function uniformly at $M \geq 2N_1$ points over the interval of definition. This is analogous to assuming M point sensors are located at the points

$$x_i = L\left(\frac{i}{M-1}\right), \quad i = 0, 1, \ldots, M-1. \tag{4}$$

Let the sampled measurement be

$$y_s(t) = [\,\overline{C}_1\,]\,\eta_1(t)$$

where $\overline{C}_1$ is the $(M \times 2N_1)$ matrix equal to (1) sampled at (4). If the pair $(A_1, \overline{C}_1)$ are used in an LQE design, the resulting estimator gain matrix is

$$\overline{K} = S\,\overline{C}_1^T\,R^{-1}$$

where S is the solution of a Riccati equation, and R is the positive definite covariance matrix in the stocastic filter. Since R is a design variable, we will take

$$R = r\,I_M, \text{ where } r \in \mathbb{R} > 0.$$

In this way, $\overline{K}$ may be expressed in the form

$$\overline{K} = \overline{S}\overline{C}_1^T, \text{ where } \overline{S} = \left(\frac{1}{r}\right)S. \tag{5}$$

After solving the system (5) for $\overline{S}$, $K(x)$ may be calculated as

$$K(x) = \overline{S}C_1^T(x).$$

Hence, the elements of $K(x)$, designed in this way, are the scaled sums of the controlled mode shapes. The design is now complete. It should be noted that the design method does not in anyway use the uncontrolled modes.

Let

$$\tilde{K}_1 = \int_0^L K(x)\, C_1(x)dx$$

and

$$\tilde{K}_2 = \int_0^L K(x)\, C_2(x)dx.$$

Then the observer may be written as

$$\frac{d}{dt}\,\hat{\eta}_1(t) = (A_1 - \tilde{K}_1)\,\hat{\eta}_1(t) + B_1u(t) + \tilde{K}_1\eta_1(t) + \tilde{K}_2\eta_2(t).$$

The closed-loop system including the uncontrolled modes is given by

$$\frac{d}{dt}\begin{bmatrix} \eta_1(t) \\ \hat{\eta}_1(t) \\ \eta_2(t) \end{bmatrix} = \begin{bmatrix} A_1 & B_1F & 0 \\ \tilde{K}_1 & A_1 - \tilde{K}_1 + B_1F & \tilde{K}_2 \\ 0 & B_2F & A_2 \end{bmatrix}\begin{bmatrix} \eta_1(t) \\ \hat{\eta}_1(t) \\ \eta_2(t) \end{bmatrix}. \qquad (6)$$

The $\tilde{K}_2$ term in (6) is the so-called observation spillover term [6]. In point sensor based control systems $\tilde{K}_2$ is non-zero, requiring the uncontrolled modes to be directly included in the design. In the design presented in this paper, this term is zero by the orthogonality of the mode shapes. That is,

$$\tilde{K}_2 = \int_0^L \overline{S}C_1^T(x)C_2(x)\, dx = 0.$$

Hence, the poles of the controlled system and the poles of the observer may be placed independently, and the poles of the uncontrolled system are left decoupled and unaffected. This decoupling is further illustrated in the following example.

III. Example

In the previous section we found that the observer spatially filters the measurements in the distributed control system to remove undesired modes. In this section we conduct a simulation experiment to compare the ability of distributed and point sensor systems to operate in the presence of uncontrolled dynamics.

Our test structure is the uniform cantilevered beam of unit mass and length L shown in Figure 1. We will consider a beam with length 5 in which the effects of the uncontrolled modes are significant. The open-loop system includes 0.5% proportional damping.

We consider two sensing methods. The first sensor is an ideal tip displacement sensor which senses the displacement of the free end. The second sensor is an ideal distributed displacement sensor which senses the displacement of the of the beam along its entire length. The first four vibrational modes are selected for the design ($N_1 = 4$). An LQG design for the distributed sensor is performed using the method described in the previous section with $M = 128$. The closed-loop poles for this 4-mode design are shown in Figure 3. Using the pole locations in Figure 3, pole placement is used to place the poles of the point-sensor observer. The controller design for the distributed-sensor may be used directly for the point-sensor system. The resulting closed-loop pole locations for the point-sensor system are identical to those shown in Figure 3. Both designs are made without considering the uncontrolled modes.

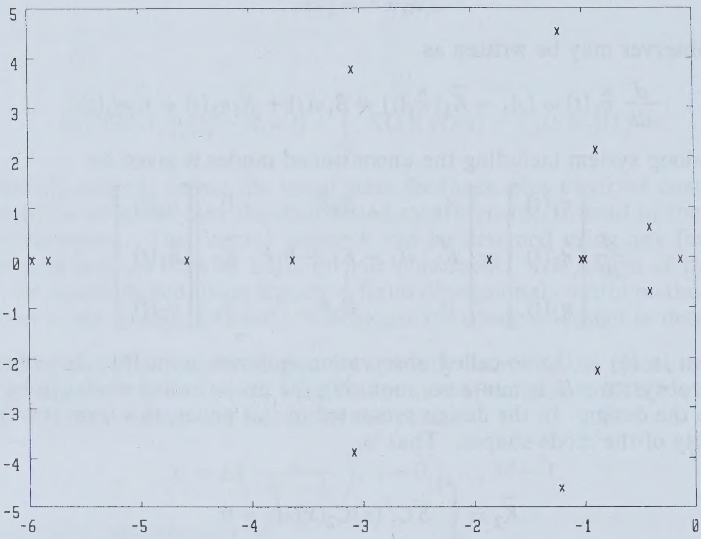


Figure 3: The closed-loop pole locations for a cantilevered beam with four vibrational modes.

The capability of the design in the presence of the uncontrolled dynamics may be tested by including some of the higher-order modes in the calculation of the closed-loop poles. If a 12-mode beam is controlled by the 4-mode distributed system ($N_2 = 8$), the resulting pole locations are shown in Figure 4. If the same 12-mode beam is controlled by the 4-mode point sensor based control system, the resulting pole locations are shown in Figure 5. Comparison of Figures 4 and 5 shows that the higher-order dynamics have little effect in the distributed case, while the poles of the point system are greatly displaced from their nominal locations, and some of the poles corresponding to the uncontrolled modes are driven unstable. This is clearly a poor design for the point sensor based system, but it illustrates how the uncontrolled modes must be considered if distributed sensing is not used. The distributed sensor based control system is insensitive to the affects of the uncontrolled modes.

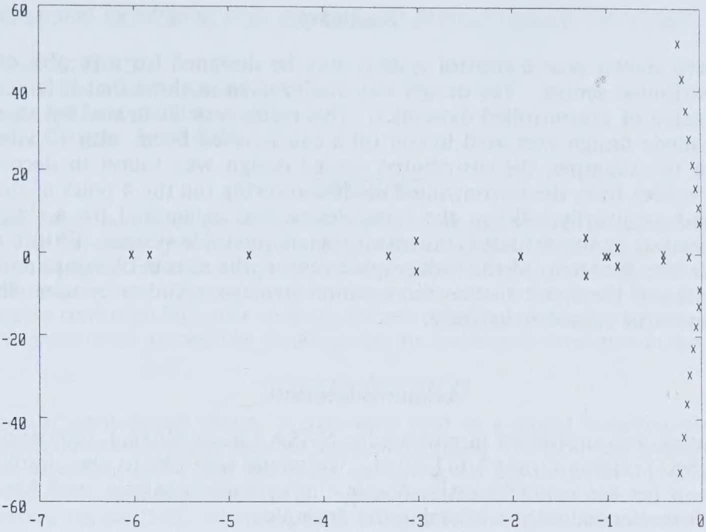


Figure 4: The closed-loop pole locations for a cantilevered beam with 12 vibrational modes controlled by the distributed sensor based control system designed to damp only four modes.

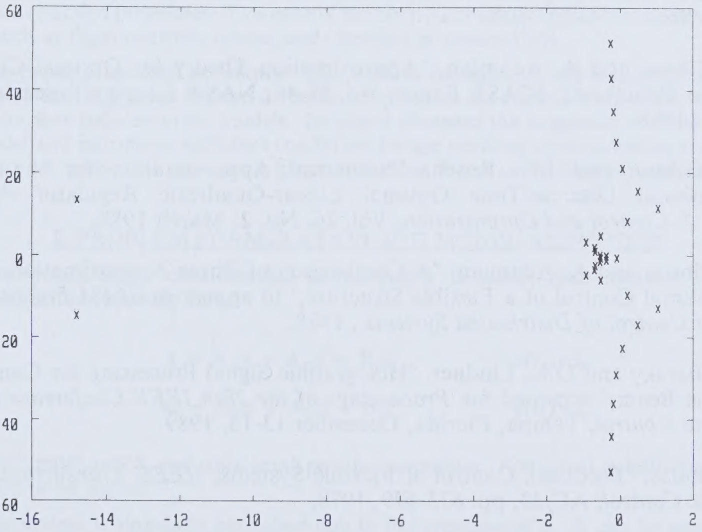


Figure 5: The closed-loop pole locations for a cantilevered beam with 12 vibrational modes controlled by the point sensor based control system designed to damp only four modes.

V. Summary

We have shown how a control system may be designed for a flexible structure using a distributed sensor. The design was analyzed to to show that it is insensitive to the inclusion of uncontrolled dynamics. This result was illustrated by an example where a 4-mode design was used to control a cantilevered beam with 12 vibrational modes. In the example, the distributed sensor design was found to decouple the controlled modes from the uncontrolled modes, allowing the the 4 poles of the design to be placed arbitrarily. When the same design was attempted for a single point sensor collocated at the actuator, the result was an unstable system. Future analysis will include the dynamics of the holographic sensor, the effects of sampling in time, and extension of the result to two dimensional structures and structures whose deflection cannot be sensed in entirety.

Acknowledgements

This work was supported in part under NASA Grant NAG-1-1006 and in part under NASA Training Grant NGT-5004. We would also like to also thank Dr. W. T. Baumann for his valuable assistance in the spillover analysis, and Mr. K. M. Reichard for independently confirming the example.

References

[1] M.F. Barsky, K.M. Reichard, D.K. Lindner, and R.O. Claus, "Distributed Sensing Methods for Controlling Flexible Structures," *Proceedings of the VPI&SU/AIAA Symposium on Dynamics and Control of Large Structures*, May 8-10, 1989.

[2] J.S. Gibson and A. Adamian, "Approximation Theory for Optimal Control of Flexible Structures," ICASE Report No. 88-48, NASA Langley Reseach Center, August 1988.

[3] J.S. Gibson and I.G. Rosen, "Numerical Approximation for the Infinite-Dimensional Discrete-Time Optimal Linear-Quadratic Regulator Problem," *SIAM J. Control and Optimization*, Vol. 26, No. 2, March 1988.

[4] J.S. Gibson and A. Adamian, "A Comparison of Three Approximation Schemes for Optimal Control of a Flexible Structure," to appear in *SIAM Frountiers Edition on Control of Distributed Systems* , 1988.

[5] M.F. Barsky and D.K. Lindner, "Holographic Signal Processing for Control of a Flexible Beam," accepted for *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, Florida, December 13-15, 1989.

[6] M.J. Balas, "Feedback Control of Flexible Systems," *IEEE Transactions on Automatic Control*, AC-23, pp. 673-679, 1978.

SLIDING MODE CONTROL FOR SINGULARLY PERTURBED SYSTEMS[1]

BONNIE S. HECK
School of Electrical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0250

ABSTRACT

This paper addresses the design of sliding mode controllers for singularly perturbed systems. The full-order system is separated into slow and fast subsystems and a sliding mode controller is designed for each subsystem. A composite of these controls is then implemented on the full-order system. The separation of the time scales in the closed loop is validated by comparing the controlled full-order model to the controlled reduced-order system. A numerical example is given which successfully demonstrates the techniques developed in this paper.

1. INTRODUCTION

Singular perturbation theory is commonly used as a model reduction technique for systems which inherently possess both fast and slow dynamics. The full-order system is separated into a fast subsystem and a slow subsystem containing the fast dynamics and slow dynamics, respectively. Much work has been done in singular perturbation theory over the last thirty years, see the survey papers [1-3]. However, until recently [4-6], singular perturbation theory was restricted for use on smooth systems, ruling out its use on systems with sliding modes.

Sliding mode theory is a control strategy where the control is discontinuous with respect to the state variables. In particular the control is piecewise-continuous with a switching surface in the state space defining the discontinuity. The control is chosen so that the system trajectories are directed towards the switching surface. Hence, once the trajectory reaches the switching surface, it is forced to move or "slide" along it. Sliding mode control has many benefits which makes it attractive to controls designers; in particular, good robustness and good disturbance rejection properties. This control has been successfully applied to numerous physical systems such as flight controls, robots, and chemical processes [7-8].

The paper is outlined as follows. The problem formulation and model reduction of the open loop system is given in Section 2. Section 3 presents the control design procedure for the fast and the slow reduced-order models. Section 4 discusses the composite control on the full-order model and introduces sufficient conditions for the resulting approximation error to be of order $O(\mu)$. A numerical example is given in Section 5 and concluding remarks are given in Section 6.

2. PROBLEM FORMULATION AND MODEL REDUCTION

The system under consideration in this paper is linear, time-invariant and can be represented in the following form:

$$\dot{x} = A_{11}x + A_{12}z + B_1u; \qquad x(t_0)=x_0 \tag{1}$$

$$\mu\dot{z} = A_{21}x + A_{22}z + B_2u; \qquad z(t_0)=z_0 \tag{2}$$

where $x \in R^p$, $z \in R^q$, $u \in R^m$, and μ is a small positive parameter. For global stability purposes, A_{22} is restricted to be stable.

This system is singularly perturbed due to the presence of μ. It can be separated into reduced-order models, one containing the fast dynamics and one containing the slow dynamics. The slow dynamic model is found by setting $\mu=0$ in equation (2). The resulting reduced-order model and an expression defining the quasi-steady-state solution, z_s, are given as follows:

---

[1] This work was supported by a Dupont Young Faculty Grant.

$$\dot{x}_s = A_0x_s + B_0u_s(x_s) \tag{3}$$

$$z_s = h(x_s) = -A_{22}^{-1}(A_{21}x_s+B_2u_s(x_s)) \tag{4}$$

where $A_0 = (A_{11} - A_{12}A_{22}^{-1}A_{21})$ and $B_0 = (B_1 - A_{12}A_{22}^{-1}B_2)$ and u_s is the slow component of the control. The quasi-steady-state solution defines a "slow manifold" in the full-order state space given by $z=h(x)$. Note that if the control is a discontinuous function of x_s, then the slow manifold is also discontinuous.

The fast dynamic model is found by introducing an expanded time variable $\tau=(t-t_0)/\mu$ and a new state variable $\eta = z-h(x)$. Equations (1) and (2) are then rewritten as:

$$\frac{d\tilde{x}}{d\tau} = \mu(A_{11}\tilde{x} + A_{12}\tilde{z} + B_1u) ; \qquad \tilde{x}(0) = x_0 \tag{5}$$

$$\frac{d\eta}{d\tau} = A_{22}\eta + B_2u_f - \frac{dh(\tilde{x})}{d\tilde{x}}\frac{d\tilde{x}}{d\tau} \qquad \eta(0) = z_0-h(x_0) \tag{6}$$

where $\tilde{z}(\tau) = z(\mu\tau+t_0)$, $\tilde{x}(\tau) = x(\mu\tau+t_0)$ and $u_f = u - u_s$. Assuming that u_f is bounded, $\tilde{x}$ stays relatively constant with respect to τ, hence, $\tilde{x}$ is approximated by x_0. Also, if $dh(\tilde{x})/d\tilde{x}$ is bounded, the expression in (6) can be simplified to obtain the resulting reduced-order model:

$$\frac{d\hat{\eta}}{d\tau} = A_{22}\hat{\eta} + B_2u_f; \qquad \hat{\eta}(0) = z_0-h(x_0) \tag{7}$$

where $\hat{\eta}$ is an approximation for η during the initial boundary layer. The fast dynamics may also be excited following a switch in the slow control. An internal boundary layer is derived by defining an expanded time variable $\tau=(t-t_1)/\mu$ and a new state variable $\eta = z-h(x)$ where t_1 is the time of the switch in the slow control. The resulting reduced-order model is the same as the model given in equation (7) except with initial conditions $\hat{\eta}(t_1) = z_s(t_1^-)-h(x_s(t_1^-))$.

3. CONTROL DESIGN

As mentioned previously, one of the most important uses of singular perturbation theory in controls design is as a model reduction technique. In particular, separate control laws can be designed for the fast dynamic model and for the slow dynamic model. The controls are then combined in a composite form for implementation on the full-order model. A sliding mode control will be developed for each of the reduced-order models in the subsections below. The validity of using singular perturbation theory in the design process will be examined by then comparing the solution of the controlled full-order model with that of the controlled reduced-order models.

Note that ideal sliding mode behavior exists when a system trajectory moves along a switching surface where the corresponding control switches with infinite frequency. In actual implementation, however, actuators cause the control to switch with high but finite speed. It is assumed in this paper that the frequency of the switching is much higher than the frequency of the fast dynamics. The assumption of ideal sliding behavior, therefore, is adequate for this analysis.

3.1 Slow Sliding Mode Control Design

The slow sliding mode control is designed using the slow reduced-order model in equation (3). The components of the control vector, $u_s(x_s)$, are chosen to have the following form:

$$u_{si}(x_s) = \begin{cases} u_{si}^+(x_s), & \text{if } s_{si}(x_s)>0 \\ u_{si}^-(x_s), & \text{if } s_{si}(x_s)\leq 0 \end{cases} \tag{8}$$

where $s_{si}(x_s)=S_{si}x_s$ is the i^th^ linear switching function, u_{si}^+ and u_{si}^- are smooth functions, and $u_{si}^+(x_s) \neq u_{si}^-(x_s)$ on the switching surface defined by $S_{si}x_s=0$. It is desired that the system slide along the intersection of the switching surfaces defined by the function $s_s(x_s) \equiv S_s x_s = 0$ where the rows of S_s are the vectors S_{si}. The equivalent control method described in reference [8] is used to obtain the sliding mode equation. The equivalent control for the slow system during sliding (denoted by the subscript *es*) is found to be given by

$$u_{es} = -(S_s B_0)^{-1} S_s A_0 x_s \tag{9}$$

Substitution of u_{es} into (3) for u_s yields the following "sliding mode equation" which describes the dynamics of the system during sliding:

$$\dot{x}_s = (A_0 - B_0(S_s B_0)^{-1} S_s A_0) x_s \tag{10}$$

This equation is further constrained by the equation $S_s x_s = 0$. The elements of the matrix S_s are chosen so that the sliding mode dynamics are stable and have desired performance.

The slow system actually slides on the surface defined by $S_s x_s = 0$ if the control components, $u_{si}^+(x_s)$ and $u_{si}^-(x_s)$, are chosen such that the system trajectory is directed toward the sliding surface from any point in a region surrounding the sliding surface, i.e. the "existence condition" [8]. The "reaching condition" states that the system trajectory is directed toward the sliding surface from any point in the state space [8]. These conditions correspond to local and global stability, respectively, of the sliding surface or equivalently, of the function s_s. Utilization of the stability concept leads to the use of a Lyapunov function in testing for the reaching condition. For the purposes of this paper, it is required that the derivative of the Lyapunov function be negative and bounded away from zero.

A suitable Lyapunov function for testing stability of a vector-valued function s is given as follows:

$$V(s) = \sum_{i=1}^{m} | s_i | \tag{11}$$

This Lyapunov function is not everywhere differentiable, therefore the Generalized Lyapunov Theorem [9] is needed which makes use of Clarke's generalized gradient. The derivative of V is defined below:

$$\dot{V}(s) = \xi^T \dot{s} \quad \text{a.e.} \qquad \forall\, \xi \in \partial V(s) \tag{12}$$

where $\partial V(s)$ is the generalized gradient of V(s) defined in [9]. From [9], $\partial V(0) = [-1,1]^m$, the unit cube in R^m and $\partial V(s) \cap (-1,1)^m = \emptyset \;\; \forall\, s \neq 0$.

If a control can be chosen such that $dV(s)/dt \leq -\epsilon < 0 \;\; \forall\, \xi \in \partial V(s)$ and for some ϵ which is not of order $O(\mu)$, then the reaching condition is satisfied. This condition can be written for the slow system, given in (3), as

$$\dot{V}(s_s) = \xi^T S_s (A_0 x_s + B_0 u_s(x_s)) \leq -\epsilon < 0 \qquad \forall\, x_s \in R^p \text{ and } \forall\, \xi \in \partial V(s_s) \tag{13}$$

where ϵ is not of order $O(\mu)$. It is assumed that a control can be chosen to satisfy this condition for this application.

3.2 Fast Sliding Mode Control Design

The fast sliding mode control is designed for the fast reduced-order model given in equation (7). The i^{th} component in the control vector is chosen in the form below

$$u_{fi}(\hat{\eta}) = \begin{cases} u_{fi}^{+}(\hat{\eta}) & \text{if } s_{fi}(\hat{\eta}) > 0 \\ u_{fi}^{-}(\hat{\eta}) & \text{if } s_{fi}(\hat{\eta}) \leq 0 \end{cases} \tag{14}$$

where $s_{fi}(\hat{\eta}) = S_{fi}\hat{\eta}$ is the i^{th} linear switching function, u_{fi}^{+} and u_{fi}^{-} are smooth functions, and $u_{fi}^{+}(\hat{\eta}) \neq u_{fi}^{-}(\hat{\eta})$ on the switching surface defined by $S_{fi}\hat{\eta} = 0$. It is desired that the system slide along the intersection of the switching surfaces, i.e. on the surface defined by $s_f(\hat{\eta}) \equiv S_f\hat{\eta} = 0$, where the rows of S_f are the vectors S_{fi}. The equivalent control for the fast system during sliding (denoted by the subscript *ef*) is given by

$$u_{ef} = -(S_fB_2)^{-1}S_fA_{22}\hat{\eta} \tag{15}$$

Substitution of u_{ef} into (7) yields the following sliding mode equation which describes the dynamics of the system during sliding:

$$\frac{d\hat{\eta}}{d\tau} = (A_{22} - B_2(S_fB_2)^{-1}S_fA_{22})\hat{\eta} \tag{16}$$

This equation is subject to the constraint that $S_f\hat{\eta} = 0$. The gradient of the switching surface, S_f, is chosen so that the sliding mode dynamics are stable and have desired performance.

The control components, $u_{fi}^{+}(\hat{\eta})$ and $u_{fi}^{-}(\hat{\eta})$, are then chosen to guarantee that the reaching condition is satisfied. As in the slow sliding mode control design, the Lyapunov function defined in (11) is used to verify the reaching condition. Again, for the purposes of this paper, the derivative of the Lyapunov function must be negative and bounded away from zero. The reaching condition is given as follows:

$$\frac{dV(s_f)}{d\tau} = \xi^T S_f(A_{22}\hat{\eta} + B_2u_f(\hat{\eta})) \leq -\epsilon < 0 \quad \forall\, \hat{\eta} \in R^q \text{ and } \forall\, \xi \in \partial V(s_f) \tag{17}$$

where ϵ is not of order $O(\mu)$. It is assumed that such a control can be found for the current application.

4. COMPOSITE CONTROL

In the actual implementation to the full-order model given in (1)-(2), the control would be a composite of the slow and fast controls as given below:

$$u = u_s(x) + u_f(z-h(x)) \tag{18}$$

where u_s is defined in (8), u_f is defined in (14) and $h(x) = -A_{22}^{-1}(A_{21}x + B_2u_s(x))$. It is interesting to note that two unrelated sliding surfaces will exist in the full-order state space, one resulting from u_f and one resulting from u_s. It must be shown that the solutions of the controlled reduced-order models approximate the solution of the full-order model with the composite control in (18). As mentioned previously, the approximation will consist of the slow model with boundary layer corrections corresponding to the initial boundary layer and internal boundary layers. Since the controlled systems are piecewise-linear, prior to hitting a switching surface the approximation error can be shown to be of order $O(\mu)$ using previous results on singular perturbation in linear systems. It remains to be shown that if the reaching conditions are met for the reduced-order systems that they are met for the full-order model and that the sliding mode equations for the reduced-order systems are good approximations of the true sliding mode equations.

4.1 Fast Time Approximation Error

To examine the sliding mode equation and the reaching conditions of the full-order model in the fast time scale, the model given in equation (6) is analyzed with the control $u_f(\eta)$ as defined in (14). The corresponding sliding surface in the full-order state-space (R^{p+q}) is defined by $s_f(\eta) = S_f\eta = 0$. Equivalently, it is the null space of $[0\ ,\ S_f]$ where the states are x and η.

The reaching condition which guarantees that the system will slide on this surface in the expanded time scale is obtained from the derivative of the Lyapunov function defined in (11):

$$\frac{dV(s_f)}{d\tau} = \xi^T S_f \frac{d\eta}{d\tau} \leq -\epsilon^* < 0 \qquad \forall\ (x,\eta) \in R^{p+q} \text{ and } \forall\ \xi \in \partial V(s_f) \tag{19}$$

where ϵ^* is not of order $O(\mu)$.

Theorem 1: Suppose that the fast control is chosen so that the reaching condition given in equation (17) is satisfied for the fast model. Then there exists a $\mu^*>0$ such that the reaching condition given in equation (19) will be satisfied for the full-order model for all $\mu \in [0,\mu^*)$ almost everywhere.

Proof: Substitution of equation (6) for $d\eta/d\tau$ in (19) yields:

$$\frac{dV(s_f)}{d\tau} = \xi^T S_f \left(A_{22}\eta + B_2 u_f(\eta) - \frac{dh(\tilde{x})}{d\tilde{x}} \frac{d\tilde{x}}{d\tau} \right) \tag{20}$$

An expression for $dh(\tilde{x})/d\tilde{x}$ obtained from the definition of h given in (4) is as follows:

$$\frac{dh(\tilde{x})}{d\tilde{x}} = -A_{22}^{-1}A_{21} - A_{22}^{-1}B_2 \frac{du_s(\tilde{x})}{d\tilde{x}} \tag{21}$$

If $\tilde{x}$ is not on a switching surface for $u_s(\tilde{x})$, then the smoothness of all u_{si}^+ and all u_{si}^- implies that $dh(\tilde{x})/d\tilde{x}$ is well-defined. If $\tilde{x}$ is on a switching surface and does slide, then $u_s(\tilde{x})$ can be replaced with the equivalent control $u_{es}(\tilde{x})$ and $dh(\tilde{x})/d\tilde{x}$ is well-defined. Therefore, for $\tilde{x}$ not on a switching surface and for $\tilde{x}$ sliding on a switching surface, $dh(\tilde{x})/d\tilde{x}$ is bounded. The case where $\tilde{x}$ is on a switching surface and the system is not sliding forms a set of measure zero.

Since the reaching condition in (17) is met for the reduced-order model for all $\hat{\eta} \in R^q$, $d\tilde{x}/d\tau$ is of order $O(\mu)$, and $dh(\tilde{x})/d\tilde{x}$ is bounded almost everywhere, there exists a $\mu^*>0$ and an $\epsilon^*>0$ such that the expression in (20) is bounded above by $-\epsilon^*$ for all $\mu \in [0,\mu^*)$ almost everywhere.■

The qualification that the reaching condition is satisfied almost everywhere does not pose a problem for the design of the sliding mode control. The reaching condition fails when $dh(\tilde{x})/d\tilde{x}$ is not bounded, i.e. when the slow control switches prior to sliding or when it switches without sliding. In that case, $dh(\tilde{x})/d\tilde{x}$ contains an impulse so that η responds with an impulse response (except if $\mu=0$ since $d\tilde{x}/d\tau = O(\mu)$). If the fast system is sliding, the impulse may force the fast dynamics off of the fast switching surface. However, since the reaching condition specified in (19) is satisfied almost everywhere, the fast dynamics will start sliding again very quickly after such a disturbance.

Theorem 1 guarantees that satisfying the reaching condition for the reduced-order model implies that the reaching condition for the full-order model is also satisfied for small enough values of μ. It remains to be shown that if the approximate solution hits its sliding surface, then the true solution will also hit its sliding surface and do so within a time difference of order $O(\mu)$, thus causing an error in the initial condition of the sliding mode equation to be of order $O(\mu)$. It must also be shown that the resulting sliding mode equations are $O(\mu)$ approximations of each other.

Corollary 1: If the reaching conditions in (19) and (17) for the full-order model and the fast model are satisfied, then the full-order model will start sliding on the surface defined by $S_f\eta=0$ (in the full-order state space) within a time difference of $O(\mu)$ (in the expanded time scale) from the time that the fast model starts sliding on the surface defined by $S_f\hat{\eta}=0$ (in the reduced-order state space).

Proof: The reaching conditions in (19) and (17) guarantee that both the fast model and the full-order model will start sliding on their respective switching surfaces in finite time, i.e., $\tau < \infty$. (See "Lyapunov's Theorem Generalized" in [9] for a proof.) Suppose the fast model starts sliding at $\tau=\tau_1$ and the full-order model starts sliding at $\tau=\tau_2$ where $\tau_2>\tau_1$. (The case where $\tau_2<\tau_1$ is easily shown using the procedure for this case.)
The following expression is utilized:

$$\Delta V(s_f(\eta)) \equiv V(s_f(\eta))\Big|_{\tau_2} - V(s_f(\eta))\Big|_{\tau_1} = \int_{\tau_1}^{\tau_2} \frac{dV(s_f(\eta))}{d\tau}\,d\tau \tag{22}$$

Note that $dV(s_f(\eta))/d\tau = dV(s_f(\hat{\eta}))/d\tau + O(\mu)$. Because the fast model is sliding, $V(s_f(\hat{\eta}))=0$ for $\tau \geq \tau_1 \Rightarrow dV(s_f(\hat{\eta}))/d\tau = 0$ for $\tau \geq \tau_1$. Therefore, the right-hand-side of (22) is of order $O(\mu)$. From (19), $\Delta V(s_f(\eta))/\Delta\tau \leq -\epsilon^* < 0$ where $\Delta\tau = \tau_2 - \tau_1$. Since ϵ^* is not of order $O(\mu)$ and $\Delta V(s_f(\eta))$ is of order $O(\mu)$, $\Delta\tau$ must be of order $O(\mu)$. ■

Corollary 2: The error between the solution of the reduced-order sliding mode equation given in (16) and the solution of the sliding mode equation of the full-order system is of order $O(\mu)$.

Proof: The sliding mode equation for the full-order model is found from equations (5) and (6) by the equivalent control method:

$$\frac{d\eta}{d\tau} = (A_{22} - B_2(S_fB_2)^{-1}S_fA_{22})\eta + B_2(S_fB_2)^{-1}S_fM\,\frac{d\tilde{x}}{d\tau} \tag{23}$$

where $M=dh(\tilde{x})/d\tilde{x}$. Because $d\tilde{x}/d\tau$ is of order $O(\mu)$ and M is bounded, the last term on the right-hand side of (23) is of order $O(\mu)$, so that (23) is a regular perturbation of (16). Since both equations are linear and the initial conditions are of order $O(\mu)$ (as verified by Corollary 1), the solutions differ by an error of order $O(\mu)$. ■

4.2 Slow Time Approximation Error

To examine the sliding mode equation and the reaching conditions resulting from the slow control for the full-order model in the normal time scale, the model given in equation (1)-(2) is analyzed. For the purposes of this section, it is advantageous to rewrite (1) and (2) into the following form where the new variable $\eta=z-h(x)$ is used.

$$\dot{x} = A_0x + A_{12}\eta + B_0u_s + B_1u_f; \qquad x(t_0) = x_0 \tag{24}$$

$$\mu\dot{z} = A_{22}\eta + B_2u_f; \qquad z(t_0) = z_0 \tag{25}$$

The slow control is $u_s(x)$ as defined in (8) where $s_{si}(x)=S_{si}x$. The corresponding sliding surface in the full-order state-space (R^{p+q}) is defined by $s_s(x)=S_sx=0$. Equivalently, it is the null space of $[S_s\ ,\ 0]$ where the states are x and z.

The reaching condition which guarantees that the full-order system will slide on the switching surface is given as:

$$\dot{V}(s_s) = \xi^T S_s\dot{x} \leq -\epsilon^* < 0 \qquad \forall\ (x,z) \in R^{p+q} \text{ and } \forall\ \xi \in \partial V(s_s) \tag{26}$$

where V(s) is defined in (11) and ϵ^* is not of order $O(\mu)$.

Theorem 2: Suppose that the slow control is chosen so that the reaching condition given in equation (13) for the slow model is satisfied. Then there exists a $\mu^*>0$ such that the reaching condition given in equation (26) for the full-order model will be satisfied for all $\mu\in[0,\mu^*)$ outside of the initial boundary layer and any internal boundary layers.

Proof: Substitution of (24) into (26) yields the following equivalent expression:

$$\dot{V}(s_s) = \xi^T S_s(A_0x + B_0u_s) + \xi^T S_s(A_{12}\eta + B_1u_f) \tag{27}$$

Suppose that the fast dynamics are designed so that sliding occurs with respect to the fast control on the surface defined by $S_f\eta=0$, then $u_f \approx u_{ef}$, the equivalent control as defined in (15). Since $\eta \rightarrow 0$ as $\tau \rightarrow \infty$, the last term on the right of (27) is of order $O(\mu)$. Satisfaction of the reaching condition in (13) for the reduced-order model implies that there exists a $\mu^*>0$ and an $\epsilon^* > 0$ such that the expression in (27) is bounded above by $-\epsilon^*$ for all $\mu\in[0,\mu^*)$. Hence, satisfaction of the reaching condition for the slow reduced-order model implies that the reaching condition is satisfied for the full-order system outside of the boundary layer for small enough values of μ. □

Theorem 2 guarantees that if the reaching condition is satisfied for the slow model then it is also satisfied for the full-order model. It can be shown that the time difference between when the full-order model starts sliding and when the slow model starts sliding is of order $O(\mu)$. It can also be shown that the resulting sliding mode equations are $O(\mu)$ approximations of each other. These proofs are similar to those in the fast time approximation, the details are omitted here due to space limitations but can be found in reference [10].

In general, the fast dynamics will be excited when the slow subsystem first hits the slow switching surface. If the reaching conditions are not met, then the slow control may switch without sliding. If the system does slide, then the slow control is generally discontinuous as it goes from $u_s^+(x)$ or $u_s^-(x)$ to $u_{es}(x)$. In either case, the switch excites the fast dynamics. The fast dynamics must then be analyzed by defining an internal boundary layer existing after the first switch in u_s. However, if the fast control u_f is designed to satisfy the reaching conditions for the fast subsystem (as discussed in Section 3.2), then the fast dynamics will start sliding again on the fast switching surface within this newly defined boundary layer and the value of η will decay rapidly. It is important to emphasize that excitement of the fast dynamics does not affect the reaching conditions or the sliding mode equation of the slow dynamics outside of this new boundary layer. Hence, this internal boundary layer *does not affect the control design process*.

An interesting aspect of the slow control analysis is that the slow manifold as defined in equation (4) is discontinuous whether the system is sliding or not. However, if sliding occurs, the value of u_s should be replaced by the equivalent control u_{es} defined in (9). This does not change the overall shape of the manifold, it simply gives a definition for the quasi-steady-state value through part, if not all, of the discontinuous region of the manifold.

5. NUMERICAL EXAMPLE

In this section, a numerical example is given which demonstrates the sliding mode control design for the slow and fast reduced-order models. The solution of the controlled full-order model is compared with the solution of the controlled reduced-order models. The system to be studied is given in the form of equation (1)-(2) where:

$$A_{11} = \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix} \quad A_{12} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad A_{21} = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix} \quad A_{22} = \begin{bmatrix} 0 & 1 \\ -18 & -6 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

where $\mu=0.1$, and $x_0=[1, 2]^T$ and $z_0=[-1, 2]^T$.

The slow sliding controller is designed for the slow model given in (3) where the parameter matrices are found to be

$$A_0 = \begin{bmatrix} -2 & 0.222 \\ 0 & -1 \end{bmatrix} \quad B_0 = \begin{bmatrix} 1.111 \\ 1 \end{bmatrix}$$

The control is chosen as follows:

$$u_s(x_s) = -(S_sB_0)^{-1}S_sA_0x_s - \delta_s(S_sB_0)^{-1}\,\mathrm{sgn}(S_sx_s) \tag{28}$$

where the switching function is defined as $s_s=S_sx_s$. (Note that sgn(z) is defined to be equal to 1 if $z>0$ and equal to -1 if $z\leq 0$.) For this application, the design parameters are chosen to be $S_s = [1, 1]$ and $\delta_s = 3$. The resulting sliding mode has an eigenvalue of -1.6. The reaching condition is tested by evaluating $dV(s_s)/dt$. In the scalar control case, $V(s_s)$ reduces to $V(s_s) = |s_s|$. Correspondingly, $dV(s_s)/dt$ is expressed as follows:

$$\dot{V}(s_s) = [\mathrm{sgn}(s_s)]^2(-\delta_s) = -\delta_s \tag{29}$$

Therefore, $dV(s_s)/dt$ is bounded above by $-\delta_s=-3$ guaranteeing that the reaching condition is met for the sliding mode in both the reduced-order model and the full-order model.

The fast control is chosen for the reduced-order model in (7). The control is chosen to be of the form below:

$$u_f(\hat{\eta}) = \left(-\alpha\sum_{i=1}^{2}|\hat{\eta}_i| - \delta_f\right)\mathrm{sgn}(S_f\hat{\eta}) \tag{30}$$

The design parameters are chosen to be $S_f = [5, 1]$, $\alpha = 1.3$ and $\delta_f = 2$. The eigenvalue of the sliding mode equation is -5. It can be shown that $dV(s_f)/d\tau$ is bounded above by $-\delta_f=-2$ for all $\hat{\eta}$, therefore, the reaching condition is satisfied for both the reduced-order model and the full-order model.

The numerical solutions of the controlled reduced-order models are compared with the numerical solutions of the controlled full-order model in Figures 1.a and 1.b. The control for the full-order model is the composite control given in (18). The reduced-order system solutions were obtained by numerically integrating the reduced-order models in (3) and (7) with the respective controls in (28) and (30). The approximation for $x(t)$ is $x_s(t)$. The approximation for $z(t)$ is $z_s(t) + \hat{\eta}(\tau)$ for $t < 0.2$ and is $z_s(t)$ for $t \geq 0.2$ since the boundary layer solution decays very rapidly. In computing z_s from equation (4), if it is determined that the system is sliding with respect to u_s, then u_s is replaced with the equivalent control u_{es} defined in (9).

As seen from these figures, the controlled reduced-order models approximate the true solution very accurately. It is important to note that two unrelated sliding surfaces exist in the full-order state space, one resulting from u_f and one resulting from u_s. In addition, the true system and the approximated system both slide on the appropriate switching surfaces and do so within a time delay of $O(\mu)$. Specifically, the fast dynamics started sliding on the surface $S_f(z-z_s(0))=0$ (in the approximated system) at $t\approx0.02$ sec and started sliding on the surface $S_f(z-h(x))=0$ (in the true system) at $t\approx0.02$ sec. The slow dynamics started sliding on the surface $S_s x_s=0$ (in the approximated system) at $t\approx1.00$ sec and started sliding on the surface $S_s x=0$ (in the true system) at $t\approx0.98$ sec. Note the fast dynamics in the true system are excited when the slow system starts sliding due to the discontinuity in the control, see Figure 1.b. For the approximation to be accurate with errors of $O(\mu)$, the fast dynamics should be evaluated after the discontinuity in the slow dynamics. However, the purpose of this example is to demonstrate the accuracy of designing sliding mode controls for the fast and slow reduced-order models and then implementing them as a composite on the full-order model. This has been shown successfully. Note that due to the presence of the fast control in the true system, the fast dynamics start sliding again within 0.03 seconds of being excited. An additional aspect of this example is that the computation time required to obtain the approximation was 1/20 of that required for the full-order model.

6. SUMMARY

In this paper, sliding mode control was investigated for use on a linear singularly perturbed system. It was determined that the separation into slow and fast subsystems was easily accomplished and greatly facilitated the control design process. In particular, separate sliding mode controllers were designed for the slow and the fast reduced-order models and then implemented as a composite on the full-order system. It should be noted that while standard sliding mode control laws may be used for the fast and the slow controls, the fast control u_f must not have magnitude on the order of $1/\mu$ prior to sliding. This restriction holds for all control laws designed using singular perturbation theory, however, sliding mode controllers tend to be designed with large "gains" to ensure that the existence or reaching condition is satisfied. Sufficient conditions were developed which guarantee that the approximation error is of order $O(\mu)$ between the solution of the controlled reduced-order models and the solution of the full-order model with a composite control. These sufficient conditions correspond to restrictions placed on the type of control law used. An interesting result of this analysis is that the use of the composite control on the full-order system causes the system to have two separate sliding modes on two separate sliding surfaces, one resulting from the slow control and one resulting from the fast control. A numerical example was presented which successfully demonstrated the techniques developed in this paper.

REFERENCES

[1] P.V. Kokotovic, R.E. O'Malley and P. Sannuti, "Singular Perturbations and Order Reduction in Control Theory--an Overview," Automatica, vol. 12, pp. 123-132, March 1976.

[2] V.R. Saksena, J. O'Reilly and P.V. Kokotovic, "Singular Perturbations and Time-Scale Methods in Control Theory: Survey 1976-1983," Automatica, vol.20, pp. 273-293, May 1984.

[3] P.V. Kokotovic, H.K. Khalil and J. O'Reilly, Singular Perturbation Methods in Control:Analysis and Design, Academic Press, London, 1986.

[4] B.S. Heck and A.H. Haddad, "Singular Perturbation in Piecewise-Linear Systems," IEEE Trans. Automat. Contr., vol. 34, pp. 87-90, Jan. 1989.

[5] B.S. Heck and A.H. Haddad, "Singular Perturbation Analysis of Linear Systems With Scalar Quantized Control," Automatica, vol. 24, pp. 755-764, Nov. 1988.

[6] B.S. Heck and A.H. Haddad, "Singular Perturbation Analysis of Linear Systems With Vector Quantized Control," Proc. of the ACC, Pittsburgh, pp. 2178-2183, June 1989.

[7] V.I. Utkin, "Variable Structure Systems with Sliding Modes," IEEE Trans. Automat. Contr., vol. 22, pp. 212-223, April 1977.

[8] V.I. Utkin, "Variable Structure Systems: Present and Future," Automation and Remote Control, vol. 44, pp. 1105-1120, Sept. 1983.

[9] B.E. Paden and S.S. Sastry, "A Calculus for Computing Filippov's Differential Inclusion with Application to the Variable Structure Control of Robot Manipulators," Proc. of the 1986 CDC, Athens, Greece, pp. 578-582 Dec. 1986.

[10] B.S. Heck, "Sliding Mode Control for Singularly Perturbed Systems," Georgia Tech Systems and Controls Report OTBH01-89.

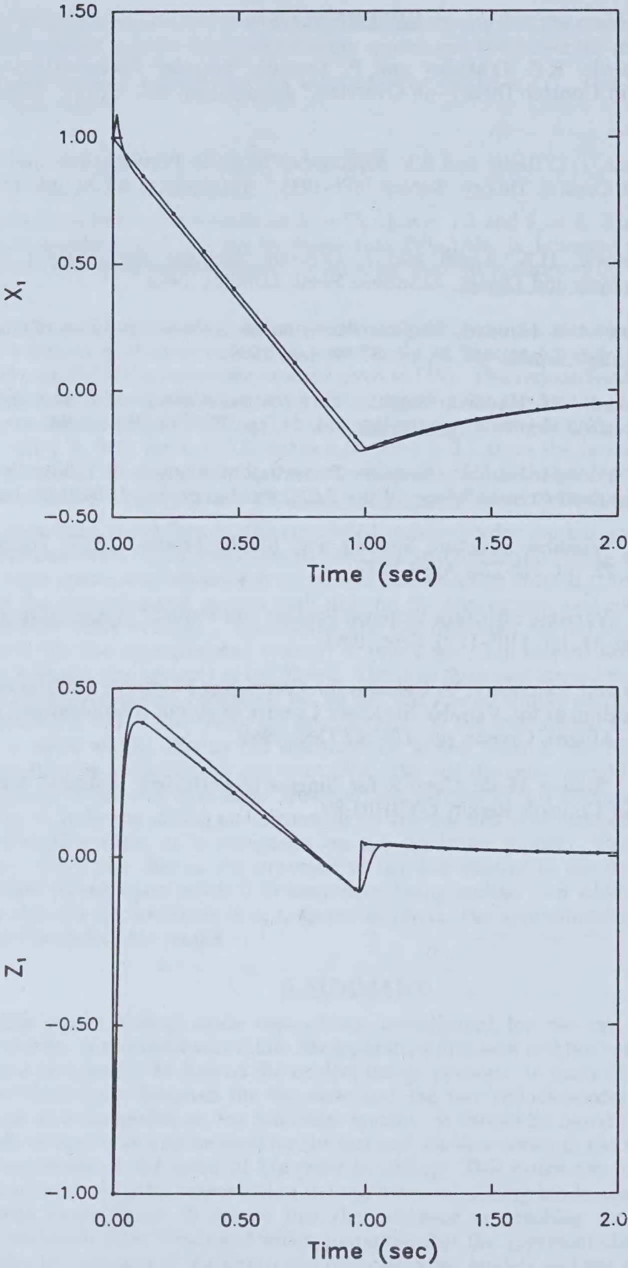


Figure 1: Response of system to initial condition for true solution (solid line) and approximate solution a) x_1 and b) z_1.

STABILITY REGIONS OF UNCERTAIN SYSTEMS WITH BOUNDED CONTROLLERS

M. MADANI-ESFAHANI, M. HACHED, AND S. H. ŻAK
School of Electrical Engineering, Purdue University, West Lafayette, IN 47907

ABSTRACT

This paper addresses the problem of estimating regions of asymptotic stability (RAS) of uncertain dynamical systems with bounded controllers and the "sliding mode" requirement. The notion of combining different Lyapunov functions along with a decoupling transformation are used in order to find estimates of the stability regions.

1. INTRODUCTION

In this paper, we investigate the problem of estimating the regions of asymptotic stability (RAS) of uncertain variable structure systems (VSS) [1] with hard bounds on the control action. The sliding mode property of VSS is desirable since it results in a robust control.

The notion of combining different Lyapunov functions is utilized. We find different Lyapunov functions, and for each Lyapunov function, we find an estimate of the RAS. Finally, we combine different regions in order to find an improved RAS. The combined region is not, in general, a convex region and cannot be found analytically by using just a single Lyapunov function.

2. STATEMENT OF THE PROBLEM

We consider a class of uncertain systems modeled by the following equations:

$$\dot{x} = Ax + B(u + \varsigma(t)) , \tag{2.1}$$

where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $u \in \mathbb{R}^m$, and $\varsigma(\cdot) : \mathbb{R} \to \mathbb{R}^m$ can be considered as lumped uncertainty. We also assume that $\varsigma(t)$ is norm bounded so that $||\varsigma(t)|| \leq \rho$, $\rho > 0$ for all $t > 0$.

We define the n-m dimensional subspace, Ω, (the switching surface) as follows:

$$\Omega \triangleq \{x : \sigma_i(x) \triangleq S_i x = 0 \ , \ \ S_i \in \mathbb{R}^{1\times n} \ \ i = 1,...,m, \ x \in \mathbb{R}^n\} . \tag{2.2}$$

Let $S = [S_1^T,...,S_m^T]^T \in \mathbb{R}^{m\times n}$, and $\sigma(x) = [\sigma_1,...,\sigma_m]^T = Sx \in \mathbb{R}^m$. We assume that S is of full rank, the matrix SB is nonsingular, and the pair (A,B) is completely controllable.

We also assume that the control u in the system (2.1) is norm bounded in the sense that $||u|| \leq K$, where $K > \rho$, and $||\cdot||$ denotes the Euclidean norm.

We shall employ a bounded switching type controller which will bring the system into the sliding mode. Then we find three different regions of asymptotic stability for the closed loop system using Lyapunov theory. Finally, we combine the obtained regions in order to improve the estimate of the region of attraction. We also show that the switching surface is attractive everywhere in the obtained regions and hence stability is achieved through the sliding mode.

3. ESTIMATES OF THE REGION OF ASYMPTOTIC STABILITY (RAS)

Utilizing the results obtained in [2], the system (2.1) can be transformed to

$$\begin{cases} \dot{z} = A_{11} z + A_{12} \sigma \\ \dot{\sigma} = A_{21} z + A_{22} \sigma + u(t) + \varsigma(t) \ , \quad \sigma(t_0) = \sigma(x(t_0)) \ . \end{cases} \tag{3.1}$$

We define $a_{ij} = ||A_{ij}||$ and $\lambda \triangleq \min\{ |\lambda_i|, i=1,...,m\}$, where $A_{11} = \mathrm{diag}\{\lambda_1,...,\lambda_{n-m}\}$ and λ_i's are the desired eigenvalues of the reduced order "slow" system which are chosen to be real and distinct. We use the following control strategy

$$u = \begin{cases} -K\dfrac{\sigma}{||\sigma||} & \text{if } \sigma \neq 0 \\ 0 & \text{if } \sigma = 0 \ . \end{cases} \tag{3.2}$$

Let $\mu = K - \rho$ and $\Sigma = \Sigma_1 \cap \Sigma_2$ where

$$\Sigma_1 \triangleq \{(z,\sigma)\colon \|\sigma\| < \frac{\lambda\mu}{a_{12}a_{21} + \lambda a_{22}} \;,\; z \in \mathbb{R}^{n-m},\; \sigma \in \mathbb{R}^m\} \,,$$

$$\Sigma_2 \triangleq \{z,\sigma)\colon a_{21}\|z\| + a_{22}\|\sigma\| < \mu,\; z \in \mathbb{R}^{n-m},\; \sigma \in \mathbb{R}^m\} \,.$$

Proposition 3.1 [2]

A trajectory that starts in Σ stays in Σ. This implies that Σ is a region of asymptotic stability (RAS).

Proposition 3.2

The region $\mathscr{R}_1$ is a domain of attraction for the system (3.1) with the control law (3.2), where

$$\mathscr{R}_1 \triangleq \{(z,\sigma)\colon \frac{a_{21}}{\lambda}\|z\| + \|\sigma\| < \frac{\lambda\mu}{\lambda a_{22} + a_{21}a_{12}}\} \,.$$

Proposition 3.3

The system (3.1) with the control (3.2) is asymptotically stable in the region $\mathscr{R}_2 \triangleq \{(z,\sigma)\colon a_{21}\|z\| + r\|\sigma\| < \mu\}$, where

$$r = \frac{a_{22}}{2} + \sqrt{(\frac{a_{22}}{2})^2 + a_{12}a_{21}} \;.$$

In the following Corollary, we combine the estimates we found in the above propositions in order to improve the region of attraction of the closed loop system.

Corollary

The region $\Sigma^* = \Sigma \cup \mathscr{R}_1 \cup \mathscr{R}_2$ is a domain of attraction for the closed loop variable structure control system (3.1), (3.2).

Proposition 3.4

The region $\Sigma_2 \triangleq \{(z,\sigma)\colon a_{21}\|z\| + a_{22}\|\sigma\| < \mu, \sigma \in \mathbb{R}^m, z \in \mathbb{R}^{n-m}, \sigma \neq 0\}$ contains points in the state space for which the condition $\sigma^T \dot{\sigma} < 0$ is satisfied.

One can show that the regions Σ and $\mathscr{R}_2$ are contained in Σ_2. Hence the stability in Σ and also in $\Sigma \cup \mathscr{R}_2$, is achieved through sliding (for details and proofs see [3]).

4. CONCLUSIONS

Using different Lyapunov functions, we found three domains of attraction for the system (2.1). Then, we combined the obtained regions in order to improve the estimated stability bounds. The improved regions do not have convex boundaries and cannot be obtained by using just a single Lyapunov function.

REFERENCES

[1] R. A. DeCarlo, S. H. Żak, and G. P. Mathews, "Variable structure control of nonlinear multivariable systems: A tutorial," Proc. of the IEEE, Vol. 76, No. 3, pp. 212-232, 1988.

[2] S. M. Madani-Esfahani, S. Hui, and S. H. Żak, "On the estimation of sliding domains and stability regions of variable structure control systems," Proc. 26th Annual Allerton Conf. on Communication, Control, and Computing, Monticello, IL, pp. 518-527, Sept. 28-30, 1988.

[3] S. M. Madani-Esfahani, M. Hached, and S. H. Żak, "Estimation of sliding mode domains of uncertain variable structure systems with bounded controllers," to appear, IEEE Transaction on Auto. Control, April 1990.

NEAR-OPTIMAL CONTROL OF DISCRETE EVENT SYSTEMS

K.M. PASSINO and P.J. ANTSAKLIS
Dept. of Electrical and Computer Engineering
University of Notre Dame, Notre Dame IN 46556

Abstract

The problem of the near-optimal control of systems accurately represented with a logical discrete event system (DES) model is formulated and solved for the deterministic case in this paper. The given DES model P characterizes the *valid* dynamical behavior of the plant and another DES model A represents design objectives which specify the *allowable DES behavior*. Under certain conditions a controller may be constructed which will select a sequence of inputs that results in allowable plant behavior. Here we consider the case where there is another part of the design objectives which indicates that the plant behavior should also, in some sense, be "near-optimal". It is within this context that we formulate a "near-optimal controller synthesis problem", i.e. how to construct a controller to achieve near-optimal allowable DES behavior. Our solution to this problem utilizes results from the theory of heuristic search to help overcome problems with computational complexity often encountered with logical DES models. This approach relies on the choice of an "evaluation function" to represent "heuristics" which are used to focus the search for an near-optimal solution. It is shown that when these heuristics over-estimate (e.g. if the heuristics are "filtered"), near-optimal solutions can still be obtained. Next, the notion of "ε-monotonicity" is introduced and used to help quantify the performance of the algorithm used to find near-optimal solutions. We also address the fact that it is, in general, quite difficult to find an appropriate heuristic function (even if it can over-estimate) for particular applications. We show that a metric space approach can be used to specify over-estimating heuristic functions in a systematic way for a wide variety of DESs. Examples are provided.

1. INTRODUCTION

This paper establishes the first steps towards developing the foundations for an near-optimal control theory for systems accurately represented with a logical discrete event system (DES) model. Here, we focus on one class of deterministic logical DES models (an automaton augmented so that it can model costs for events to occur) and the case where complete state information is available. (These results are based on those established in [7,8].) We utilize the DES controller synthesis methodology introduced in [7]. The technique requires viewing the plant model P as characterizing the *valid DES behavior* while another DES model A represents design objectives which specify the *allowable* (desirable) *DES behavior* which is "contained in" the valid behavior. Here, we are concerned with those times when the design objectives dictate that not only should the DES behavior be allowable but that it is in some sense near-optimal. To quantify this, a performance index is defined in terms of the costs for the events to occur. The near-optimal controller synthesis problem for deterministic DES (defined in Section 2) consists of the construction of a controller that chooses a sequence of inputs that will transfer the DES from its initial state to one state in a set of final states and produce a near-optimal state trajectory.

Our solution to this near-optimal control problem utilizes approaches developed in the theory of heuristic search to help overcome problems with computational complexity often encountered in the study of logical DESs. The idea is to *search* for a near-optimal solution among all the possible solutions to the near-optimal controller synthesis problem. In Section 3, we outline the main results from the theory of the widely used A^* algorithm adapted to the near-optimal controller synthesis problem. It is shown how even if over-estimating heuristics are used, near-optimal ("ε-optimal") solutions are found by A^*. Utilizing a general formulation we explain the approaches used in the literature for finding ε-optimal solutions. Then we show several new techniques to "filter the heuristics" (dynamic weighting) which will still result in an ε-optimal solution. Next, we introduce ε-monotonicity and show that it is equivalent to "ε-consistency" and that it automatically implies "ε-admissibility". It is proven that if A^* uses an ε-monotone heuristic then it will find ε-optimal state paths to all states expanded. The heuristic search approach relies on the choice of a "heuristic function" which is used to focus the search for an near-optimal solution. The problem one encounters though is that it is, in general, quite difficult to find an ε-admissible and ε-monotone heuristic function for many applications. In Section 3.3 we show that the metric space approach used in [7] can be extended and used to specify an ε-admissible and ε-monotone heuristic function in a systematic way for a wide variety of DESs. Hence, the results offer a computationally efficient approach to solving the near-optimal controller synthesis problem. To illustrate the results we provide a solution to a near-minimum-input/event cost problem in Section 4 and study ε-monotone heuristics for the case where it is known that the event costs lie in a certain region. Comparisons to relevant research are made throughout the paper.

2. THE NEAR-OPTIMAL CONTROLLER SYNTHESIS PROBLEM FOR DETERMINISTIC DESs

We consider DESs that can be accurately modelled with

$$P=(X,Q,\delta,\chi,x_0,X_f) \tag{1}$$

where

(i) X is the possibly infinite set of plant states,
(ii) Q is the finite set of plant inputs (controller outputs),
(iii) $\delta:Q\times X\rightarrow X$ is the plant state transition function,
(iv) $\chi:X\times X\rightarrow \mathbb{R}^+$ is the *event cost function*,
(v) x_0 is the initial plant state, and
(vi) $X_f\subset X$ is the non-empty finite set of *final states*.

The model P is limited to representing DESs that are deterministic in the sense that for a given input there is exactly one possible next state. The set

$$E(P)=\{(x,x')\in X\times X\text{: } x'=\delta(q,x)\}\cup\{(x_d,x_0)\} \tag{2}$$

denotes the (possibly infinite) set of events for our plant P (x_d is a dummy state, and (x_d,x_0) a dummy event added for convenience). The event cost function $\chi(x,x')$ is defined for all $(x,x')\in E(P)$; it specifies the "cost" for each event (state transition) to occur and it is required that there exist a $\delta'>0$ such that $\chi(x,x')\geq\delta'$ for all $(x,x')\in E(P)$. The *discrete event controller* (DEC) is $C=(Q,X,\xi,q_0)$ where (i) Q is the finite set of controller states (plant inputs), (ii) X is the set controller inputs (plant outputs), (iii) $\xi:X\times Q\rightarrow Q$ is the controller transition function, and (iv) q_0 is the initial controller state. The closed loop *discrete event control system* (DECS) is formed by connecting the outputs of the plant (states) to the inputs of the controller and the outputs of the controller to the inputs of the plant. The state of the DECS is given by certain pairs $(q,x)\in Q\times X$. The pair (q_0,x_0) is a valid DECS state where q_0 and x_0 are the initial states of the controller and plant respectively. If (q,x) is a valid state for the DECS then a valid next state is given by (q',x') where $q'=\xi(x,q)$ and $x'=\delta(q',x)$.

Mathematical Preliminaries: State Trajectories and (x,X_z)-Reachability

Let Z be an arbitrary set. Z^* denotes the set of all finite strings over Z including the empty string Ø. For any $s,t\in Z^*$ such that $s=zz'\cdots z''$ and $t=yy'\cdots y''$, st denotes the concatenation of the strings s and t, and $t\in s$ is used to indicate that t is a substring of s, i.e., $s=zz'\cdots t\cdots z''$. For brevity, the notation $s_{zz''}$ is used to denote a string $s\in Z^*$ such that $s=zz'\cdots z''$ begins with the element $z\in Z$ and ends with $z''\in Z$. Let z_0 be a distinguished member of the set Z. The notation s_z is used to denote a string $s\in Z^*$ such that $s=z_0z'\cdots z$ begins with z_0 and ends with $z\in Z$. Furthermore, $s_{z>}$ denotes a string $s\in Z^*$ such that $s=zz'z''\cdots$ begins with $z\in Z$ and the end element is not specified. A (finite) *cycle* is a string $s\in Z^*$ such that $s=zz'\cdots z''z$ has the same first and last element $z\in Z$. A string $s\in Z^*$ is *cyclic* if it contains a cycle (for $t_{zz}\in Z^*$, $t_{zz}\in s$), and *acyclic* if it does not. Let $|s|$ for $s\in Z^*$ denote the length of string $s\in Z$, i.e., the number of elements of Z concatenated to obtain s.

A string $s \in X^*$ is called a *state trajectory* or *state path* of P if for all successive states $xx' \in s$, $x' = \delta(q,x)$ for some $q \in Q$. Let

$$E_s(P) \subset E(P) \tag{3}$$

denote the set of all events needed to define a particular state path $s \in X^*$ that can be generated by P. For some state path $s = xx'x''x''' \cdots$, $E_s(P)$ is found by simply forming the pairs (x,x'), (x',x''), (x'',x'''), $\cdots$. An *input sequence* $u \in Q^*$ that produces a state trajectory $s \in X^*$ is constructed by concatenating $q \in Q$ such that $x' = \delta(q,x)$ for all $xx' \in s$. Let $X_z \subset X$ and

$$\mathfrak{X}(P,x,X_z) \subset X^* \tag{4}$$

denote the set of all finite state trajectories $s = xx' \cdots x''$ of P beginning with $x \in X$ and ending with $x'' \in X_z$. Then, for instance, $\mathfrak{X}(P,x_0,X_f)$ denotes the set of all finite length state trajectories for P that begin with the initial state x_0 and end with a final state $x \in X_f$.

A plant P is said to be *(x,X_z)-reachable* if there exists a sequence of inputs $u \in Q^*$ that produces an acyclic state trajectory $s \in \mathfrak{X}(P,x,X_z)$. Then, for instance, if P is (x_0,X_f)-reachable we know that there exists an acyclic state trajectory $s = x_0 x \cdots x'$ with $x' \in X_f$.

Allowable DES Behavior

The *valid behavior* that the DES can exhibit which is modelled by P can be characterized by the set of all its valid state trajectories $\mathfrak{X}(P,x,X_f)$ where $x \in X$, along with its input sequences (it is specified with the graph of P). Let $P = (X,Q,\delta,\chi,x_0,X_f)$ specify the valid behavior of the plant and

$$A = (X_a,Q_a,\delta_a,\chi_a,x_{a0},X_{af}) \tag{5}$$

be another DES model which we think of as specifying the "allowable" behavior for the plant P. Allowable plant behavior must also be valid plant behavior. Formally, we say that the allowable plant behavior described by A *is contained in* P, denoted with A[P], if the following conditions on A are met:

(i) $X_a \subset X$,
(ii) $Q_a \subset Q$,
(iii) $\delta_a: Q_a \times X_a \to X_a$ is given by $\delta_a(q,x) = \begin{cases} \delta(q,x) & \text{if } \delta(q,x) \in X_a \\ \text{undefined} & \text{otherwise} \end{cases}$
(iv) $\chi_a: X_a \times X_a \to \mathbb{R}^+$ is a restriction of $\chi: X \times X \to \mathbb{R}^+$,
(v) $x_{a0} = x_0$,
(vi) $X_{af} \subset X_f$.

Also, let $E(A) \subset E(P)$ denote the set of *allowable events* defined as in (2). The model A, specified by the designer, represents the "allowable" DES plant behavior which is contained in the valid DES behavior described by the given P. It may be that entering some state, using some input, or going through some sequence of events is undesirable. Such design objectives relating to what is "permissible" or "desirable" plant behavior are captured with A. This formulation is similar in character to the "supervisor synthesis problem" formulated in a language-theoretic framework and solved in [15,16]. There the authors use languages to specify the "acceptable" (allowable) and "legal" (valid) DES behavior. They develop an algorithm to synthesize a supervisor which will make the behavior of the supervisory control system characterized by yet another language to "lie between" the acceptable and legal languages.

The Near-Optimal Controller Synthesis Problem

The *performance index*

$$J: X_a^* \to \mathbb{R}_+ \tag{6}$$

is defined in terms of the costs of the events by

$$J(s) = \sum_{(x,x')\in E_s(A)} \chi(x,x') \tag{7}$$

for all $s\in \mathfrak{X}(A,x,X_a)$ where $x\in X_a$. By definition, $J(s)=0$ if $s=x$ where $x\in X_a$. As in conventional near-optimal control the objective is to find an input sequence that will produce a state trajectory s such that J(s) is close to its minimum value. Here, we are particularly interested in state trajectories that end in X_{af}. Let $s^*_{x>}\in \mathfrak{X}(A,x,X_{af})$ denote an *optimal state trajectory*, then

$$J(s^*_{x>})=\inf\{J(s_{x>}):s_{x>}\in \mathfrak{X}(A,x,X_{af})\} \tag{8}$$

where $x\in X_a$. Since the graph of A is locally finite there are only a finite number of state trajectories of finite length; hence $J(s^*_{x>})=\min\{J(s_{x>}):s_{x>}\in \mathfrak{X}(A,x,X_a)\}$ where $x\in X_a$. There may, in general, be more than one state path where the minimum is achieved. Let $X_z\subset X_a$. The set of minimum cost state paths for A, beginning at state $x\in X_a$, and ending at state $x'\in X_z$ is denoted by $\mathfrak{X}^*(A,x,X_z)\subset\mathfrak{X}(A,x,X_z)$; consequently, $\mathfrak{X}^*(A,x_0,X_{af})$ denotes the set of optimal allowable state trajectories that begin with x_0 and end in X_{af}. A state trajectory $s\in \mathfrak{X}(A,x,X_{af})$ will be said to be "near-optimal" or "ε-optimal" if

$$J(s)\leq(1+\varepsilon)J(s^*) \tag{9}$$

for $s^*\in \mathfrak{X}^*(A,x,X_{af})$ where $\varepsilon\geq 0$. Notice that s and s^* may not end in the same state.

The Near-Optimal Controller Synthesis Problem (NCSP)

> Let A describe the allowable behavior for a plant P such that A[P]. Assume that A is (x_0,X_{af})-reachable where x_0 is the initial plant state. Find a controller that will generate a sequence of inputs that drives A along a state trajectory $s\in \mathfrak{X}(A,x_0,X_{af})$ such that $J(s)\leq(1+\varepsilon)J(s^*)$ where $J(s^*)=\min\{J(s):s\in \mathfrak{X}(A,x_0,X_{af})\}$ with $\varepsilon\geq 0$.

Once an ε-optimal state trajectory s is found, an input sequence, say $u\in Q^*$ can be constructed to drive P along s. From this a controller C can be constructed. It is possible that some general conventional near-optimal control formulations could be adapted for the above problem but the particular formulation above lends itself to a computationally efficient solution, allows for the specification of, for instance, a "near-minimum-input/event cost" control problem, and is quite useful in applications.

3. NEAR-OPTIMAL CONTROLLER SYNTHESIS VIA HEURISTIC SEARCH

In this section we solve the problem of how to find an ε-optimal allowable state trajectory in A beginning at x_0 and ending in X_{af} assuming that A is (x_0,X_{af})-reachable and A[P] for a given plant P. The approach here, as in [7], is to use the **A*** algorithm.

3.1 The A* Algorithm [3,6,10]

The A* algorithm is one of the most widely used heuristic search algorithms. It utilizes information about how promising it is that particular state paths are on an optimal state trajectory to reduce the computational complexity. Such information is referred to collectively as "heuristic information". The heuristic information is quantified with the "evaluation function". Since the appropriate information is not available to compute $J(s^*)=J(s^*_x)+J(s^*_{xx'})$ A* estimates $J(s^*)$ with some easily computable *evaluation function* given by

$$\hat{f}:X^*_a\rightarrow\mathbb{R}_+ \tag{10}$$

which is defined for all $s\in X^*_a$ such that $s\in \mathfrak{X}(A,x,X_a)$ where $x\in X_a$. The value of $J(s^*_x)$ is estimated using

$$\hat{g}:X^*_a\rightarrow\mathbb{R}_+ \tag{11}$$

where $\hat{g}(s_x)=J(s_x)$ for all $s_x\in \mathfrak{X}(A,x_0,X_a)$. To estimate $J(s^*_{xx'})$, the remaining cost to be incurred from state x to some final state $x'\in X_{af}$, the function

$$\hat{h}:X_a\rightarrow\mathbb{R}_+ \tag{12}$$

is used with $\hat{h}(x)=0$ if $x\in X_{af}$. The function $\hat{h}$ is called the "heuristic function" since it provides the facility for supplying the **A*** algorithm with special information about the particular search problem under consideration to focus the search of **A***. The proper choice of the heuristic function can result in efficient search. The evaluation function is often chosen to be $\hat{f}(s_x)=\hat{g}(s_x)+\hat{h}(x)$ where $x\in X_a$ is the current state considered by the **A*** algorithm. Here we shall consider several other possible choices for $\hat{f}$ and show that these choices result in **A*** returning an optimal or near-optimal solution. The evaluation function must take on a certain form as we now discuss. Let s_x and s'_x denote two state trajectories that end in state $x\in X_a$, i.e., $s_x,s'_x\in \mathcal{X}(A,x_0,x)$. Let $\bar{s}\in \mathcal{X}(A,x,X_a)$. The evaluation function $\hat{f}(s)$ must be chosen so that it is "order preserving", i.e., that $\hat{f}(s_x)\geq\hat{f}(s'_x) \Rightarrow \hat{f}(s_x\bar{s})\geq\hat{f}(s'_x\bar{s})$. This requirement is a version of the principle of optimality that states that if a path s_x is judged to be more meritous that another path s'_x then no common extension $\bar{s}$ of s_x or s'_x can reverse the judgement [1,10].

The **A*** algorithm proceeds by generating candidate state trajectories which are characterized with two sets $C\subset E(A)$ and $O\subset E(A)$. The contents of C and O change at different stages of the algorithm but it is always the case that there does not exist $(x^1,x^2)\in C\cup O$ and $(x^3,x^4)\in C\cup O$ such that $x^2=x^4$ and $x^1\neq x^3$. Let the set of state trajectories of A, investigated by **A***, be denoted by $\mathcal{X}(A,C,O)$. Each state path $s_{x'}\in \mathcal{X}(A,C,O)$ begins with x_0, the initial state, and has an end state $x'\in X_a$ such that $(\cdot,x')\in C\cup O$. For $s,s'\in X_a^*$ let $s\leftarrow ss'$ denote the operation of *replacing* s by ss'. To find $s_{x'}\in \mathcal{X}(A,C,O)$ from C and O choose $(x,x')\in C\cup O$ and let $s=xx'$. Repeat the following steps until x_d is encountered: (a) Find $(x^1,x^2)\in C\cup O$ with $x^2=x$ where $s=x\cdots$, (b) Let $s\leftarrow x^1s$, and go to (a). The operation of finding the set $\mathcal{E}(x)=\{x':x'\in X_a \text{ and } x'=\delta_a(q,x)\}$ is called *expanding the state* $x\in X_a$. For Z and Z' arbitrary sets let $Z\leftarrow Z'$ denote the *replacement* of Z by Z'.

```
A*:
(1) Let $C=\{\}$ and $O=\{(x_d,x_0)\}$.
(2) If $|O|>0$, then go to Step 3. If $|O|=0$, then exit with no solution.
(3) Choose $(x,x')\in O$ so that $\hat{f}(s_xx')$ is a minimum (resolve ties arbitrarily).
    Let $O\leftarrow O-\{(x,x')\}$ and $C\leftarrow C\cup\{(x,x')\}$.
(4) If $x'\in X_{af}$ then exit with $s_{x'}\in \mathcal{X}^*(A,x_0,X_{af})$, a (near-) optimal state trajectory.
(5) For each $x''\in \mathcal{E}(x')$:
    (i) If for all $\bar{x}\in X_a$, $(\bar{x},x'')\notin C\cup O$ then let $O\leftarrow O\cup\{(x',x'')\}$.
    (ii) If there exists $\bar{x}\in X_a$ such that $(\bar{x},x'')\in O$ and $\hat{f}(s_{x'}x'')<\hat{f}(s_{\bar{x}}x'')$ then let $O\leftarrow O-\{(\bar{x},x'')\}$ and $C\leftarrow C\cup\{(x',x'')\}$.
    (iii) If there exists $\bar{x}\in X_a$ such that $(\bar{x},x'')\in C$ and $\hat{f}(s_{x'}x'')<\hat{f}(s_{\bar{x}}x'')$then let $C\leftarrow C-\{(\bar{x},x'')\}$ and $O\leftarrow O\cup\{(x',x'')\}$.
```

A* is said to be *complete* since it terminates with a solution. A heuristic function $\hat{h}(x)$ is said to be *admissible* if $0\leq\hat{h}(x)\leq J(s^*_{x>})$ for all $x\in X_a$ such that $s^*_{x>}\in \mathcal{X}^*(A,x,X_{af})$. Let **A**$^*(\hat{h}(x))$ denote an **A*** algorithm which uses $\hat{h}(x)$ as its heuristic function. If $\hat{h}(x)$ is admissible then **A**$^*(\hat{h}(x))$ is said to be *admissible* since it is guaranteed to find an optimal state trajectory when one exists, i.e., when A is (x_0,X_{af})-reachable. A heuristic function $\hat{h}(x)$ is said to be *monotone* if $\hat{h}(x)\leq\chi(x,x')+\hat{h}(x')$ for all $(x,x')\in E(A)$. A heuristic function $\hat{h}(x)$ is said to be *consistent* (equivalent to being monotone) if $\hat{h}(x)\leq J(s^*_{xx'})+\hat{h}(x')$ for all $(x,x')\in E(A)$ where $s^*_{xx'}\in \mathcal{X}^*(A,x,x')$. If $\hat{h}(x)$ a monotone heuristic function then **A**$^*(\hat{h}(x))$ finds optimal paths to all expanded states, i.e., $\hat{g}(s_x)=J(s^*_x)$ for all $x\in X_a$ with $(\cdot,x)\in C$, $s_x\in \mathcal{X}(A,C,O)$, and $s^*_x\in \mathcal{X}^*(A,x_0,x)$. The real utility of knowing that $\hat{h}(x)$ is monotone lies in the fact that states are expanded at most once. This implies that the **A*** algorithm can be simplified by removing Step 5 (iii) since events (pointers) will never be taken from C and placed in O.

3.2 The $\mathbf{A^*}$ Algorithm for Finding Near-Optimal Solutions

In this section we consider the evaluation function

$$\hat{f}_\varepsilon(s_x)=\alpha_g\hat{g}(s_x)+\hat{h}(x) \tag{13}$$

with $0<\alpha_g\leq 1$ which appears similar to the standard $\mathbf{A^*}$ algorithm except we shall, at times, allow the heuristic function to overestimate the remaining cost to find a solution.

The heuristic function $\hat{h}(x)$ is said to be *ε-admissible* if

$$\hat{h}(x)\leq(1+\varepsilon)J(s_{x>}^*) \tag{14}$$

for all $x\in X_a$ and $s_{x>}^*\in\mathfrak{X}^*(A,x,X_{af})$ where $\varepsilon\geq 0$. $\mathbf{A^*}$ is said to be ε-admissible if it returns an ε-optimal solution (state trajectory).

Theorem 1: If $\hat{h}(x)$ is ε-admissible and $0<\alpha_g\leq 1$ then $\mathbf{A^*}(\hat{h}(x))$ finds a state trajectory $s\in\mathfrak{X}(A,x_0,X_{af})$ such that $J(s)\leq(1+\varepsilon)J(s^*)/\alpha_g$ where $s^*\in\mathfrak{X}^*(A,x_0,X_{af})$.

Proof: The fact that $\mathbf{A^*}$ terminates with a solution is proven in a manner similar to the way termination is proven in [3,10]. Assume the opposite, i.e., that $\mathbf{A^*}$ terminates with $x_f\in X_{af}$ so that $\hat{f}_\varepsilon(s_{x_f})>(1+\varepsilon)J(s^*)/\alpha_g$ for $s_{x_f}\in\mathfrak{X}(A,x_0,X_{af})$ and $s^*\in\mathfrak{X}^*(A,x_0,X_{af})$. When x_f was chosen for expansion, for each $(\cdot,x)\in O$ it satisfied $\hat{f}_\varepsilon(s_{x_f})\leq\hat{f}_\varepsilon(s_x)$ where $s_x\in\mathfrak{X}(A,C,O)$. But notice that an optimal path s^* is assumed to exist. For this particular path at any step before termination there must exist at least one state, say x' such that $x'\in s^*$ such that $(\cdot,x')\in O$. Split s^* at state x', i.e., let $s^*=s_{x'}^*s_{x'>}^*$. Also, choose x' so that $|s_{x'}^*|$ is of minimum length, i.e. so that x' is the first state on s^* such that $(\cdot,x')\in O$. Then all the ancestors of x', say x_a have $(\cdot,x_a)\in C$; therefore $\hat{g}(s_{x'}^*)=J(s_{x'}^*)$. Using the assumption of ε-admissibility $\hat{f}_\varepsilon(s_{x'}^*)=\alpha_g\hat{g}(s_{x'}^*)+\hat{h}(x')\leq(1+\varepsilon)J(s_{x'}^*)+(1+\varepsilon)J(s_{x'>}^*)=(1+\varepsilon)J(s^*)$. Hence, before termination there will always exist a state $x'\in s^*$ such that $(\cdot,x')\in O$ and $\hat{f}_\varepsilon(s_{x'}^*)\leq(1+\varepsilon)J(s^*)$; hence, $\hat{f}_\varepsilon(s_{x_f})>(1+\varepsilon)J(s^*)/\alpha_g$. This is a contradiction so $\mathbf{A^*}$ terminates with $x_f\in X_{af}$ returns s_{x_f} such that $J(s_{x_f})\leq(1+\varepsilon)J(s^*)/\alpha_g$. ■

This result is proven in [9] for the case of $\alpha_g=1$. A slightly different approach is used in [2] where the author introduces a "bandwidth constraint" for the heuristic function and shows that $\mathbf{A^*}$ will return an near-optimal solution. Let

$$z_h:X_a\rightarrow\mathbb{R}_+ \tag{15}$$

with $0\leq z_h(x)\leq(1+\varepsilon)$ denote what will be called a "$\hat{h}$-filter" that is used to weight the heuristics. Using Theorem 1 we can overview the results in the literature on finding ε-optimal solutions with $\mathbf{A^*}$. First, consider the case where $\hat{g}$ and $\hat{h}$ are weighted but $\mathbf{A^*}$ still returns an optimal solution. Choosing $z_h(x)=\omega$ for all $x\in X_a$ and $\alpha_g=(1-\omega)$ with $0\leq\omega\leq\frac{1}{2}$ results in an $\mathbf{A^*}(\hat{h}(x))$ that is admissible provided $\hat{h}(x)=z_h(x)\hat{h}'(x)$ with $\hat{h}'(x)$ admissible. This choice was first considered by Pohl in [12] where he showed that ω can be varied to emphasize the breadth-first (ω small) or depth-first (ω large) components of $\mathbf{A^*}$'s search. The following Corollary to Theorem 1 provides another method to weight the components of the evaluation function.

Corollary 1: If $z_h(x)=\alpha_h$ where $0\leq\alpha_h\leq\alpha_g$ and $\hat{h}(x)=z_h(x)\hat{h}'(x)$ where $\hat{h}'(x)$ is admissible then $\mathbf{A^*}(\hat{h}(x))$ is admissible.

This result generalizes Pohl's weighting scheme in [12] by showing how the weights can be set so that optimality can still be obtained. For the remainder of the paper we focus on the case where $\mathbf{A}^*$ returns ε-optimal solutions and $\alpha_g=1$.

Corollary 2: If $\hat{h}(x)=z_h(x)\hat{h}'(x)$ where $\hat{h}'(x)$ is an admissible heuristic function and $\alpha_g=1$ then $\mathbf{A}^*(\hat{h}(x))$ is ε-admissible.

Using Corollary 2 we first outline the results in the literature. Let $d(x)$ denote the depth of state x (i.e., $d(x)=|s_x|$) and N be an upper bound on the depth of the state at the end of the longest path investigated by $\mathbf{A}^*$ before termination; hence, $0\leq d(x)\leq N$. In [13,14] the author used $z_h(x)=(1+\varepsilon(1-d(x)/N))$ ("dynamic weighting") and $\alpha_g=1$ to obtain a search that will begin with a strong depth-first component and finish with a strong breadth-first component to search. In [9] the authors generalized the results in [13,14] by showing that $\alpha_g=1$ and $z_h(x)=(1+\varepsilon z_\rho(x))$, where $z_\rho(x)$ is defined so that $0\leq z_\rho(x)\leq 1$ will result in $\mathbf{A}^*$ returning an ε-optimal solution. Corollary 2 slightly generalizes Pearl and Kim's result by showing (i) the full range of values that the filter $z_h(x)$ can take on and still guarantee ε-admissibility, and (ii) that $\hat{g}(s_x)$ can also be weighted with α_g. In [9] the authors also introduce another approach to ε-optimal search via a more complex algorithm than $\mathbf{A}^*$ called $\mathbf{A}^*_\varepsilon$. They argue that the advantage of $\mathbf{A}^*_\varepsilon$ over the approach in [13,14] is that it allows for loading heuristic information into another heuristic function and that N does not have to be known a priori. (For a comparison of $\mathbf{A}^*_\varepsilon$ to Pohl's dynamic weighting approach see [9,10].) Next, we show how the flexibility in choosing the $\hat{h}$-filter also allows the designer to load heuristic information into the evaluation function and no a priori information such as N is needed.

We introduce several candidate $\hat{h}$-filters $z_h(x)$ for the case where $\alpha_g=1$. We focus on the case where it increases the computational efficiency of $\mathbf{A}^*$ to alternate between emphasis on a breadth-first component to a depth-first component of search. Such situations occur quite often when there is a portion of the graph which is more efficiently searched with one or the other of these techniques. Suppose, for instance, that at some depth N_s it is most beneficial to switch from breadth-first to depth-first search (or vice versa) then the following $\hat{h}$-filters can be utilized:

(1) $z_h(x)=(1+\varepsilon)\left[\frac{d(x)}{N_s+d(x)}\right]$ (for switching from breadth-first to depth-first)

(2) $z_h(x)=(1+\varepsilon)\left[\frac{N_s}{N_s+d(x)}\right]$ (for switching from depth-first to breadth-first)

Another possibility is that certain regions of the graph of A can be more efficiently searched with different emphasis on breadth or depth-first search. In this case partition X_a into sets X_{ai} for $i=1,2,\ldots n$. Then, let $z_h(x)=z_h(x')$ for all $x,x'\in X_{ai}$ for each $i=1,2,\ldots n$. This will allow for a switching in the emphasis between breadth and depth-first search as the algorithm expands states in different parts of the graph of A.

Next, we introduce a new condition on $\hat{h}(x)$ which we call "ε-monotonicity" and study several properties of $\mathbf{A}^*$ algorithms that operate with ε-monotone heuristics. The heuristic function $\hat{h}(x)$ is said to be *ε-monotone* if $\hat{h}(x)\leq(1+\varepsilon)\chi(x,x')+\hat{h}(x')$ for all $(x,x')\in E(A)$. The heuristic function $\hat{h}(x)$ is said to be *ε-consistent* if $\hat{h}(x)\leq(1+\varepsilon)J(s^*_{xx'})+\hat{h}(x')$ for all $x,x'\in X_a$ where $s^*_{xx'}\in\mathfrak{X}^*(A,x,x')$.

Theorem 2: If $\hat{h}(x)$ is ε-consistent then it is ε-admissible ($\hat{h}(x)$ is ε-monotone iff it is ε-consistent).

Proof: For the first part of the Theorem let $s^*_{xx'}\in\mathcal{X}^*(A,x,X_{af})$ for any $x\in X_a$, then $\hat{h}(x')=0$ and $\hat{h}(x)\leq(1+\varepsilon)J(s^*_{xx'})$ for all $x\in X_a$. For the second part clearly if $\hat{h}(x)$ is ε-consistent it is ε-monotone. The proof for the other direction follows by repeated application of the ε-monotone condition along an optimal path $s^*_{xx'}\in\mathcal{X}^*(A,x,x')$. ■

Theorem 3: If $\hat{h}(x)$ is ε-monotone then $A^*(\hat{h}(x))$ finds ε-optimal paths to all expanded states, i.e., $\hat{g}(s_x)\leq(1+\varepsilon)J(s^*_x)$ for all $x\in X_a$ with $(\cdot,x)\in C$, $s_x\in\mathcal{X}(A,C,O)$, and $s^*_x\in\mathcal{X}^*(A,x_0,x)$.

Proof: Assume that $A^*(\hat{h}(x))$ selects a state x for expansion so that $\hat{g}(s_x)>(1+\varepsilon)J(s^*_x)$. If $(\cdot,x)\in O$ is the only pair in O then all ancestors of x have been expanded and $\hat{g}(s_x)=J(s^*_x)$. The assumption that $\hat{g}(s_x)>(1+\varepsilon)J(s^*_x)$ implies that s^*_x contains at least one more state $x'\in s^*_x$ such that $(\cdot,x')\in O$. Let $s^*_x=s^*_{x'}s^*_{x'x}$ where x' minimizes $|s^*_{x'}|$. Now, it will be shown that x' and not x (as assumed above) should be selected for expansion. From the proof of Theorem 1 $\hat{g}(s_{x'})=J(s^*_{x'})$ and since $\hat{h}(x)$ is assumed ε-consistent

$$\hat{f}_\varepsilon(s_{x'})=\hat{g}(s_{x'})+\hat{h}(x')\leq(1+\varepsilon)J(s^*_{x'})+(1+\varepsilon)J(s^*_{x'x})+\hat{h}(x)=(1+\varepsilon)J(s^*_x)+\hat{h}(x)$$

The assumption $\hat{g}(s_x)>(1+\varepsilon)J(s^*_x)$ gives $\hat{f}_\varepsilon(s_{x'})<\hat{f}_\varepsilon(s_x)$ so x' will be expanded which is a contradiction. Therefore, $\hat{g}(s_x)\leq(1+\varepsilon)J(s^*_x)$ for any state x expanded. ■

The approach used to prove Theorems 1, 2, and 3 is similar to the approaches used to prove the analogous results in standard A^* theory [3,6,10]. Unfortunately, unlike the case for monotone heuristics, we cannot guarantee that Step 5 (iii) of the A^* algorithm can be removed; consequently it is possible that A^* will re-expand states that have already been expanded. Since, $\hat{h}(x)-\hat{h}(x')\leq(1+\varepsilon)\chi(x,x')$, for relatively small ε we expect A^* to focus its search nearly as well as the case where A^* operates with a monotone heuristic function.

3.3 Specification of the Heuristic Function

There has been extensive work on the problem of how to automatically generate heuristics for an arbitrary problem. These techniques have been outlined in [7]. Although all these results are somewhat relevant, none directly answer the question of how to find a heuristic function that will be ε-admissible and ε-monotone. This is the problem that is addressed here. We show that the approach developed in [7] can be easily extended so that ε-admissible and ε-monotone heuristics can be specified for a wide variety of DESs. As in [7] the approach taken here relies on the use of metric spaces. Let Z be an arbitrary non-empty set and let $\rho:Z\times Z\rightarrow\mathbb{R}$ where ρ has the following properties: (i) $\rho(x,y)\geq0$ for all $x,y\in Z$ and $\rho(x,y)=0$ iff $x=y$, (ii) $\rho(x,y)=\rho(y,x)$ for all $x,y\in Z$, and (iii) $\rho(x,y)\leq\rho(x,z)+\rho(z,y)$ for all $x,y,z\in Z$ (triangle inequality). The function ρ is called a *metric* on Z and $\{Z;\rho\}$ is a *metric space*. Let $p\in Z$ and define $d(p,Z)=\inf\{\rho(p,z):z\in Z\}$. The value of $d(p,Z)$ is called the *distance between point p and set* Z. Let $\Delta(Z)$ denote the class of functions that are metrics on the set Z.

Theorem 4: For the DES P and A[P] if $\hat{h}(x)=\inf\{\rho(x,x_f):x_f\in X_{af}\}$ and $\rho\in\Delta(X_a)$ with $\rho(x,x')\leq(1+\varepsilon)\chi(x,x')$ for all $(x,x')\in E(A)$ then $\hat{h}(x)$ is ε-admissible and ε-monotone.

Proof: For ε-monotonicity let $s_{\bar{x}x''}\in \mathfrak{X}(A,\bar{x},A_{af})$ where $\bar{x}\in X_a$ and let $xx'\in s_{\bar{x}x''}$ be two successive states on $s_{\bar{x}x''}$. Notice that for the sequence of states $x\in X_a$ expanded, the state at which the inf is achieved in $\hat{h}(x)=\inf\{\rho(x,x_f):x_f\in X_{af}\}$ may change. Let x_p denote the state at which the inf is achieved for x and x_p' the one for x'. By the triangle inequality, $\rho(x,x_p')\leq\rho(x,x')+\rho(x',x_p')$. But by the definition of $\hat{h}(x)$ we know that $\rho(x,x_p)\leq\rho(x,x_p')$. It follows that $\rho(x,x_p)\leq\rho(x,x')+\rho(x',x_p')$. By the definition of $\hat{h}(x)$ we have $\hat{h}(x)\leq\rho(x,x')+\hat{h}(x')$ and since $\rho(x,x')\leq(1+\varepsilon)\chi(x,x')$, $\hat{h}(x)\leq(1+\varepsilon)\chi(x,x')+\hat{h}(x')$ for all $x,x'\in X_a$ such that $xx'\in s$ where $s\in\mathfrak{X}(A,\bar{x},X_{af})$ which guarantees the ε-monotonicity of $\hat{h}(x)$. From Theorem 2, ε-monotonicity implies ε-admissibility so the proof is done. ■

We have still not said how to specify the exact form of the heuristic function for particular applications. There is a wide class of DESs whose state space can be modelled in terms of $X\subset\mathbb{R}^n$. As evidence of this fact we turn to the many applications of the theory of Petri nets [11] (e.g. General or Extended Petri nets) where the states are n-tuples of natural numbers or the more recent work in [4] where numerical n-tuples are used. It is easy to specify a wide variety of metrics on $\mathbb{R}^n$ [5]; hence most often there is no problem in finding a heuristic function for P.

4. EXAMPLES

In this section we briefly outline two examples which illustrate some of the above results. First, we consider ε-optimal search when it is known that the event costs for P defined by (1) satisfy $\alpha\leq\chi(x,x')\leq\beta$ for all $(x,x')\in E(A)$. This occurs quite frequently in applications. Suppose that via the results in [7] and Theorem 4 we choose a heuristic function $\hat{h}(x)=\inf\{\rho(x,x_f):x_f\in X_{af}\}$ with $\rho\in\Delta(X_a)$ and $\rho(x,x')\leq\beta$ for all $(x,x')\in E(A)$. In this case we are allowing the heuristic function to over-estimate. The near-optimality of the solution returned depends on α and β. The values of ε such that $(1+\varepsilon)\chi(x,x')\geq\beta$ will result in an ε-monotone $\hat{h}(x)$. Hence, provided $\varepsilon\geq(\beta-\alpha)/\alpha$ we have a ε-monotone heuristic so A^* will return an ε-optimal solution. Intuitively, if α is close to β, ε can be chosen to be small and a solution more near the optimal one is found. If $\beta-\alpha$ is large then the original choice for $\hat{h}(x)$ results in a heuristic that will over-estimate by large amounts and solutions farther away from an optimal one are found.

Next, we study the *near-minimum-input/event cost control* of deterministic DESs. Consider a DES modelled with P as defined in (1) except we shall define χ in a special manner. Let $\chi_i:Q\rightarrow\mathbb{R}^+$ and require that there exists a $\delta'>0$ such that $\chi_i(q)\geq\delta'$ for all $q\in Q$ and let $\chi_e:X\times X\rightarrow\mathbb{R}_+$ be defined for all $(x,x')\in E(P)$. The function χ_i specifies the *cost of each input* and χ_e specifies the cost of each event (state transition) as in (1). For all $q\in Q$, $x,x'\in X$ such that $x'=\delta(q,x)$ define $\chi(x,x')=\chi_i(q)+\chi_e(x,x')$. The function χ then specifies the combined cost of the input and event. Given A such that A[P] and $A^*(\hat{h}(x))$ where $\hat{h}(x)$ is ε-admissible will result in an ε-optimal solution to the NCSP.

We shall discuss different approaches to specifying the heuristic function $\hat{h}(x)$ and illustrate the relationship between ε-monotonicity and monotonicity for this example. At times it is easier to specify a heuristic function in terms of χ_e (rather than χ) since there is a close correspondence between the distance $\rho(x,x')$ between two states x and x' and $\chi_e(x,x')$. Suppose that this is the case and that via Theorem 4, $\hat{h}(x)=\inf\{\rho(x,x_f):x_f\in X_{af}\}$ with $\rho\in\Delta(X_a)$ and $\rho(x,x')\leq(1+\varepsilon)\chi_e(x,x')$ for all $(x,x')\in E(A)$ so that $\hat{h}(x)$ is ε-monotone relative to the costs

$\chi_e(x,x')$. If the input costs satisfy $\chi_i(q)\geq\varepsilon\chi_e(x,x')$ for $\varepsilon>0$ where $x'=\delta(q,x)$ then $\rho(x,x')\leq\chi_e(x,x')+\chi_i(q)$; hence, $\hat{h}(x)$ is monotone. This example shows how to specify a monotone heuristic via an ε-monotone one. It also shows that an optimal path (in terms of the costs $\chi(x,x')$) will be returned by $\mathbf{A}^*$ but an ε-optimal path in terms of the costs $\chi_e(x,x')$ is returned. Another possibility is to use the results of [7] to find a heuristic function $\hat{h}(x)=\inf\{\rho(x,x_f): x_f\in X_{af}\}$ with $\rho\in\Delta(X_a)$ and $\rho(x,x')\leq\chi(x,x')$ for all $(x,x')\in E(A)$ so that $\hat{h}(x)$ is monotone. Notice that in this case if there exists $\varepsilon>0$ such that $\chi_i(q)\leq\varepsilon\chi_e(x,x')$ whenever $x'=\delta(q,x')$ then $\rho(x,x')\leq(1+\varepsilon)\chi_e(x,x')$ for all $(x,x')\in E(A)$ so that $\hat{h}(x)$ is ε-monotone relative to the costs $\chi_e(x,x')$ (similar to the above result). Work continues on illustrating the results in this paper on physical examples.

Acknowledgment: The authors gratefully acknowledge the partial support of the Jet Propulsion Laboratory.

References

[1] Bellman R., Dynamic Programming, Princeton Univ. Press, NJ, 1957.

[2] Harris L.R., "The Heuristic Search Under Conditions of Error", Artificial Intelligence, Vol. 5, pp. 217-234, 1974.

[3] Hart P.E., Nilsson N.J., Raphael B.,"A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Trans. on Systems Science and Cybernetics, Vol. SSC-4, No. 2, pp. 100-107, July 1968.

[4] Li Y., Wonham W.M., "A State-Variable Approach to the Modelling and Control of Discrete-Event Systems", Proc. of the 26th Allerton Conf. on Communication, Control, and Computing, pp. 1140-1149, Univ. of Illinois at Champaign-Urbana, Sept. 1988.

[5] Michel A.N., Herget C.J., Mathematical Foundations in Engineering and Science: Algebra and Analysis, Prentice-Hall, NJ, 1981.

[6] Nilsson N.J., Principles of Artificial Intelligence, Tioga Pub., NY, 1980.

[7] Passino K.M., Antsaklis P.J., "On the Optimal Control of Discrete Event Systems", To appear in the Proc. of the Conf. on Decision and Control, Tampa Florida, Dec. 1989.

[8] Passino K.M., Analysis and Synthesis of Discrete Event Regulator Systems, Ph.D. Dissertation, Dept. of Electrical and Computer Eng., Univ. of Notre Dame, April 1989.

[9] Pearl J., Kim J.H., "Studies in Semi-Admissible Heuristics", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 4, pp. 392-399, July 1982.

[10] Pearl J., Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, Reading, Mass., 1984.

[11] Peterson J.L., Petri Net Theory and the Modeling of Systems, Prentice Hall, NJ, 1981.

[12] Pohl I., "Heuristic Search Viewed as Path Finding in a Graph", Artificial Intelligence, Vol. 1, pp. 193-204, 1970.

[13] Pohl I. "Practical and Theoretical Considerations in Heuristic Search Algorithms", in E.W. Elcock, D. Michie ed., Machine Intelligence 8, Ellis Horwood, NY, 1977.

[14] Pohl I., "The Avoidance of (relative) Catastrophe, Heuristic Competence, Genuine Dynamic Weighting and Computational Issues in Heuristic Problem Solving", Proc. of the 3rd Int. Joint Conf. on Artificial Intelligence, pp. 12-17, 1973.

[15] Ramadge P.J., Wonham W.M., "Supervisory Control of a Class of Discrete Event Processes", SIAM J. Control and Optimization, Vol. 25, No. 1, Jan. 1987.

[16] Wonham W.M., Ramadge P.J., "On the Supremal Controllable Sublanguage of a Given Language", SIAM J. Control and Optimization, Vol. 25, No. 3, May 1987.

SUPERVISORY CONTROL OF VECTOR DISCRETE-EVENT SYYTEMS

N.Q. Huang,* Y. Li and W.M. Wonham

Systems Control Group
Department of Electrical Engineering
University of Toronto
Toronto, Ont. M5S 1A4, Canada

September 22, 1989

Abstract

The supervisory control of a structured discrete-event system (DES) called vector DES is studied. The legal behavior specification is given in the form of a vector DES, and the "on-line" computation of the optimal supervisory control is derived; the "closed-form" expression of this optimal control is also considered.

1 Introduction

A *Vector Discrete-Event System* (VDES) is described by a 4-tuple [3]:

$$\mathbf{G} = (\mathbf{X}, \Sigma, \delta, X_0),$$

where $\mathbf{X} = \mathbf{Z}^n = \{[x_1, x_2, \cdots, x_n] \,|\, x_i \in \mathbf{Z}\}$ is the state space, $X_0 \in \mathbf{X}$ is the initial state, Σ is the set of event labels, and $\delta : \Sigma \times \mathbf{X} \to \mathbf{X}$ is the (partial) transition function. We write $\delta(\alpha, X)!$ if $\delta(\alpha, X)$ is defined, *i.e.* α can occur at X. In this paper, δ is described as follows:

$$\delta(\alpha, X) = \begin{cases} X + E_\alpha & \text{if } X + E_\alpha \geq 0 \\ undefined & \text{otherwise.} \end{cases}$$

Here $E_\alpha \in \mathbf{X}$ is the *displacement vector* for α.

Thus VDES has a structured state space, and a structured transition function. We exploit such structural information to obtain computationally efficient solutions to various control problems. This is the motivation for studying VDES [3].

Let $\Sigma = \Sigma_c \bigcup \Sigma_u$, the union of controllable-event set Σ_c and the uncontrollable-event set Σ_u. Assume $\Sigma_c \bigcap \Sigma_u = \emptyset$. We extend δ to be defined on $\Sigma^* \times \mathbf{X}$ as

$$\delta(\epsilon, X) := X \qquad \text{for the empty string } \epsilon$$

$$\delta(s\alpha, X) := \delta(\alpha, \delta(s, X))$$

if δ(s,X)! and $\delta(\alpha, \delta(s, X))!$. We define the language of $\mathbf{G}$ and the reachable set of $\mathbf{G}$, respectively, as:

$$L(\mathbf{G}) = \{s | s \in \Sigma^*, \delta(s, X_0)!\}$$

$$R(\mathbf{G}) = \{X | s \in L(\mathbf{G}), \delta(s, X_0) = X\}.$$

---

*Visiting Scholar from Suzhou University, Suzhou, China

A *predicate* P on **X** is a subset of **X**. A *linear predicate* is given as

$$P = \{X \in \mathbf{X} | AX \leq b\},$$

where A $= [a_1, a_2, \cdots, a_n]$ for some $a_i, b \in \mathbf{Z}$. Let $\Gamma = \{\gamma | \gamma \in 2^{\Sigma}, \Sigma_u \subseteq \gamma\}$ be the admissible control set of **G**. We call the function f: $\mathbf{X} \rightarrow \Gamma$ the *state feedback* of **G**. The controller f: $\mathbf{X} \rightarrow \Gamma$ is static in the sense that its control action depends only on the current plant state and not on how this state is reached. Applying f to **G** results in the closed loop system $f/\mathbf{G}$:

$$f/\mathbf{G} = (\mathbf{X}, \Sigma, \delta^f, X_0),$$

with

$$\delta^f(\alpha, X) = \begin{cases} \delta(\alpha, X) & \text{if } f(\alpha, X) = 1 \text{ and } \delta(\alpha, X)! \\ undefined & \text{otherwise.} \end{cases}$$

One basic problem is to construct , for a given predicate P, a state feedback f^* such that $f^*/\mathbf{G}$ behaves as freely as possible without violating P, *i.e.*

$$R(f^*/\mathbf{G}) = sup\{R(f/\mathbf{G}) \subseteq P | \text{f is a state feedback control}\}.$$

This problem was studied in [4], where the computation of f^* is reduced to solving a linear integer programming problem (LIP) by exploiting system structure.

Continuing the investigation along this line, we study the control of VDES with the legal-behavior specification given in the form of another VDES. Let **E** be a VDES:

$$\mathbf{E} = (\mathbf{Y}, \Sigma, \xi, Y_0)$$

with displacement vectors $H_\alpha (\alpha \in \Sigma)$. Define the *synchronous product*, or *composition*, of **E** and **G** as

$$\mathbf{E}||_c\mathbf{G} = (\mathbf{X} \times \mathbf{Y}, \Sigma, \delta \times \xi, X_0 \times Y_0)$$

where $\delta \times \xi$ is described as

$$(\delta \times \xi)(\alpha, X \times Y) = \begin{cases} (X, Y) + (E_\alpha, H_\alpha) & \text{if } \delta(\alpha, X)! \,\&\, \xi(\alpha, Y)! \\ undefined & \text{otherwise.} \end{cases}$$

It is easy to prove

$$s \in L(\mathbf{E}||_c\mathbf{G}) \Leftrightarrow s \in L(\mathbf{E}) \cap L(\mathbf{G}),$$

or

$$L(\mathbf{E}||_c\mathbf{G}) = L(\mathbf{E}) \cap L(\mathbf{G}).$$

In this paper we use **E** to describe the desired, or legal system behavior, and call $L(\mathbf{E}||_c\mathbf{G})$ the *legal language.* A *supervisory control* of **G** is a map **V**: L(**G**) → Γ. So **V**(s) specifies the events enabled by a supervisor after s. The pair (**G**,**V**) will be written **V**/**G**, reading "**G** under the supervision of **V**". We use L(**V**/**G**) defined below to describe the behavior of the closed-loop system **V**/**G** [6]:

(i) $\epsilon \in L(\mathbf{V}/\mathbf{G})$;

(ii) If $s \in L(\mathbf{V}/\mathbf{G})$, $\alpha \in \mathbf{V}(s)$ and $s\alpha \in L(\mathbf{G})$ then $s\alpha \in \mathrm{L}(\mathbf{V}/\mathbf{G})$;

(iii) No other strings belong to L(**V**/**G**).

Another basic problem is to find for a given specification **E** (in VDES form) the supremal supervisory control **V*** of **E**, *i.e.* to find **V*** such that $L(\mathbf{V}^*/\mathbf{G})$ is as large as possible without violating the legal specification:

$$L(\mathbf{V}^*/\mathbf{G}) = sup\{L(\mathbf{V}/\mathbf{G}) \subseteq L(\mathbf{E}||_c\mathbf{G}) | \text{Vis a supervisory control of } \mathbf{G}\}.$$

We call a language K $\subseteq \Sigma^*$ *controllable wrt* **G** [6] if

$$\bar{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \bar{K}.$$

The definition of L(V/G) implies that L(V/G) is controllable wrt **G**. From [6] we know the converse is also true. That is, for any controllable language K there is a supervisory control **V** of **G** such that

$$L(\mathbf{V}/\mathbf{G}) = K.$$

If we define

$$C(\mathbf{E}) = \{K | K \subseteq L(\mathbf{E}\|_c\mathbf{G}) \ \& \ K \ is\ controllable\},$$

then the problem is to find $\mathbf{V}^*$ satisfying the equation

$$L(\mathbf{V}^*/\mathbf{G}) = supC(\mathbf{E}).$$

We discuss in the next section how to compute $\mathbf{V}^*$ in an "on-line" fashion: given $s \in L(\mathbf{G})$, compute $\mathbf{V}^*(s)$. Section 3 presents the condition under which a closed form of $\mathbf{V}^*$ can be obtained, *i.e.* the control function $\mathbf{V}^*$ can be modeled after a VDES. An example is provided in Section 5.

2 "On Line" Control of VDES

In this section we discuss VDES control with a specification given in VDES form. The VDES control with specification given in the form of a linear predicate on the state space has been studied in [4], where it was shown that if **G** is *loop-free* wrt Σ_u then the computation of an optimal feedback control function enforcing the linear specification can be reduced to solving a standard linear integer programming problem (LIP). The concept of loop-freedom plays a central role in the development of [4]. It turns out that this concept is also essential to the proofs of the main results in this paper. One main result provided in this section states that if the plant **G** and the specification **E** are both loop-free wrt Σ_u then the computation of $\mathbf{V}^*(s)$ can be reduced to solving n' LIPs, n' being the dimension (or the number of state variables) of the VDES **E**.

Definition 1. We say a VDES **G** has a *loop* wrt $\Sigma' \subseteq \Sigma$ if there exist $\alpha_1, \alpha_2, \cdots, \alpha_p \in \Sigma'$ and $n_1, n_2, \cdots, n_p \in \{1, 2, \cdots, m\}$ such that

$$E_{\alpha_1}(n_1) < 0, E_{\alpha_2}(n_2) < 0, \cdots, E_{\alpha_p}(n_p) < 0$$

$$E_{\alpha_1}(n_2) > 0, E_{\alpha_2}(n_3) > 0, \cdots, E_{\alpha_p}(n_1) > 0.$$

If **G** has no loop wrt Σ' then we say **G** is *loop-free wrt* Σ'.

For $s \in$L(G)$\cap\Sigma'^*$, let $|s|_\sigma$ =number of σ in s. If X=$\delta(s, X_0)$! then

$$X = X_0 + \sum_{\sigma\in\Sigma'} n_\sigma E_\sigma \geq 0. \tag{1}$$

Here $n_\sigma = |s|_\sigma$. The reverse may not be true: the condition (1) may not imply that there exists $s, |s|_\sigma = n_\sigma$ for $\sigma \in \Sigma'$ such that $\delta(s, X_0)$!. It was proved in [4] that the reverse holds if **G** is loop-free wrt Σ'^*. We state the result in the following theorem.

Theorem 1 If **G** is loop-free wrt Σ', then for every set of integers $\{n_\sigma \geq 0 | \sigma \in \Sigma'\}$

$$X + \sum_{\sigma\in\Sigma'} n_\sigma E_\sigma \geq 0 \ \Rightarrow \ (\exists l \in \Sigma'^*) n_\sigma = |l|_\sigma \wedge \delta(l, X)!.$$

□ Using this result we can prove

Proposition 1 If **G** and **E** are loop-free wrt Σ_u, then for $s \in L(\mathbf{E}) \cap L(\mathbf{G})$:

$$(\forall l \in \Sigma_u^*) sl \in L(\mathbf{G}) \Rightarrow sl \in L(\mathbf{E}) \tag{2}$$

if and only if for all sets of integers $\{n_\sigma \geq 0 | \sigma \in \Sigma_u\}$,

$$X + \sum_{\sigma\in\Sigma_u} n_\sigma E_\sigma \geq 0 \Rightarrow Y + \sum_{\sigma\in\Sigma_u} n_\sigma H_\sigma \geq 0, \tag{3}$$

where $X = \delta(s, X_0)$, $Y = \xi(s, Y_0)$.

□

Using Proposition 1 we now show that when the control specification is given as an n'-dimensional VDES **E** which is loop-free wrt Σ_u, the optimal supervisory control can be computed by solving n' LIPs; these programming problems have the same feasible solution set but different objective functions.

Let the specification be given by VDES **E**. If L(**E**$||_c$**G**) is controllable wrt **G** then $\mathbf{V}^*(s)$ can be computed trivially: for $\sigma \in \Sigma_c$, $\sigma \in \mathbf{V}^*(s)$ if and only if $s\sigma \in L(\mathbf{E})$. We now discuss how to compute $\mathbf{V}^*$ when $L(\mathbf{E}||_c\mathbf{G})$ is not controllable. Basically, what we will do is to disable a controllable event α after the event string s if there exists a string of uncontrollable events which is feasible after α in **G** but not in **E**:

$$(\exists l \in \Sigma_u^*)\ s\alpha l \in L(\mathbf{G}) \text{ but } s\alpha l \notin L(\mathbf{E}).$$

We provide an algorithm to compute $\mathbf{V}^*(s)$ for a given s:

(i) $\Sigma_u \subseteq \mathbf{V}^*(s)$;

(ii) For $\alpha \in \Sigma_c$, if for all sets of integers $\{n_\sigma \geq 0 | \sigma \in \Sigma_u\}$

$$X + E_\alpha + \sum_{\sigma\in\Sigma_u} n_\sigma E_\sigma \geq 0 \Rightarrow Y + H_\alpha + \sum_{\sigma\in\Sigma_u} n_\sigma H_\sigma \geq 0$$

then $\alpha \in \mathbf{V}^*(s)$; here $X := \delta(s, X_0)$, $Y := \xi(s, Y_0)$.

(iii) No other events are in $\mathbf{V}^*(s)$.

Theorem 2 Assume that **G**, **E** are loop-free wrt Σ_u. If supC(**E**)$\neq \emptyset$, then the $\mathbf{V}^*$ defined above is the optimal supervisory control enforcing **E**, *i.e.* $\mathbf{V}^*$ satisfies:

$$L(\mathbf{V}^*/\mathbf{G}) = supC(\mathbf{E}). \tag{4}$$

□

From Theorem 2, we can compute $\mathbf{V}^*$(s) as follows. To decide if $supC(\mathbf{E}) \neq \emptyset$ we have to decide whether

$$min_{X_0+\sum_{\sigma\in\Sigma_u} n_\sigma E_\sigma \geq 0}(Y_0 + \sum_{\sigma\in\Sigma_u} n_\sigma H_\sigma) \geq 0$$

That is, we have to solve the following n' LIPs:

$$min(Y_0(j) + \sum_{\sigma\in\Sigma_u} n_\sigma H_\sigma(j))$$

$$s.t. \begin{cases} X_0 + \sum_{\sigma\in\Sigma_u} n_\sigma E_\sigma \geq 0 \\ n_\sigma \geq 0, \qquad n_\sigma \ \ integer \end{cases}$$

for j=1, $\cdots$, n', with n' being the dimension of **E**; and then check that the solutions to these n' LIPs are all nonnegative or not. If they are all nonnegative then $supC(\mathbf{E}) \neq \emptyset$; otherwise $supC(\mathbf{E}) = \emptyset$.

To decide if $\alpha \in \mathbf{V}^*(s)$, for $\alpha \in \Sigma_c$, we have to decide whether

$$min_{X+\sum_{\sigma\in\Sigma_u} n_\sigma E_\sigma \geq 0}(Y + \sum_{\sigma\in\Sigma_u} n_\sigma H_\sigma) \geq 0$$

Here $X := \delta(s\alpha, X_0)$, $Y := \xi(s\alpha, Y_0)$. That is, we have to solve the following n' LIPs:

$$min(Y(j) + \sum_{\sigma\in\Sigma_u} n_\sigma H_\sigma(j))$$

$$s.t. \begin{cases} X + \sum_{\sigma \in \Sigma_u} n_\sigma E_\sigma \geq 0 \\ n_\sigma \geq 0, \qquad n_\sigma \ \ integer \end{cases}$$

for j=1, $\cdots$, n', with n' being the dimension of **E**; and then check that the solutions to these n' LIPs are all nonnegative or not. If they are all nonnegative then $\alpha \in \mathbf{V}^*(s)$; otherwise $\alpha \notin \mathbf{V}^*(s)$.

3 Closed Form of Supervisory Control

As we indicated before, for a given specification **E** we can construct the optimal supervisory control $\mathbf{V}^*$ of **G** satisfying:

$$L(\mathbf{V}^*/\mathbf{G}) = supC(\mathbf{E}).$$

Generally, we cannot find a closed form expression of $\mathbf{V}^*$, *i.e.* a VDES **S**, such that

$$L(\mathbf{S}||_c\mathbf{G}) = L(\mathbf{V}^*/\mathbf{G}) = supC(\mathbf{E})$$

(We have counter-example). But if **G** and **E** have certain special structure, then this is achievable. As before, let plant $\mathbf{G} = (\mathbf{X}, \Sigma, \delta, X_0)$, and the specification $\mathbf{E} = (\mathbf{Y}, \Sigma, \xi, Y_0)$. The following two assumptions capture the desirable special structure of **G** and **E**.

A1. **G** and **E** are loop-free wrt Σ_u.

A2 . The VDES **G** satisfies the following condition: for all $\alpha \in \Sigma_u$, E_α has at most one negative element, and this negative element, if any, is -1.

From the two assumptions we can prove that there exists a VDES **S** such that $L(\mathbf{S}||_c\mathbf{G}) = supC(\mathbf{E})$. We first introduce some notation. Let **x**,**y** be the state variables of **G** and **E** respectively. Let x_0 and y_0 be the initial values of **x**,**y**. Let $E_\alpha(\mathbf{x})$ be the **x** component of E_α. Bring in the following notation:

$$\begin{aligned} \alpha^- &= \{\mathbf{x}|E_\alpha(\mathbf{x}) < 0\} \\ \bar{\alpha}^- &= \{\mathbf{y}|H_\alpha(\mathbf{y}) < 0\} \\ \mathbf{x}^- &= \{\alpha|E_\alpha(\mathbf{x}) > 0\} \\ \mathbf{x}^+ &= \{\alpha|E_\alpha(\mathbf{x}) < 0\} \\ \mathbf{x}^\Sigma &= \mathbf{x}^- \cup \mathbf{x}^+. \end{aligned}$$

We can extend the above notation to the case of multiple state variables or events. For instance, for a set S of state variables we define

$$S^- := \bigcup \{\mathbf{x}^- | \mathbf{x} \in S\}.$$

We call α^- the *post-state-variable set* of α (from **A2** if $\alpha \in \Sigma_u$, it is a singleton), $\mathbf{x}^-$ the *post-event set* of **x**.

For a given **E** if for every α in Σ_u, $\bar{\alpha}^- = \emptyset$ then L(**E**) is controllable wrt **G**. If there exists α in Σ_u such that $\bar{\alpha}^- \neq \emptyset$ but $\alpha^- = \emptyset$ then $supC(\mathbf{E}) = \emptyset$ (in this case α can always happen in **G** but cannot always happen in **E**). For $\alpha \in \Sigma_u$ with $\bar{\alpha}^- \neq \emptyset \,\&\, \alpha^- = \{\mathbf{x}\} \neq \emptyset$ we define the *operator* T_α (from VDESs to VDESs) as

$$\mathbf{E}_{new} = T_\alpha \mathbf{E} = (\mathbf{Y}, \Sigma, \xi_{new}, Y_{0,new})$$

with corresponding $Y_{0,new}$ and $H_{\sigma,new}$ being defined as follows:

(i)

$$y_{0,new} = \begin{cases} y_0 + x_0 H_\alpha(\mathbf{y}) & \text{if } \mathbf{y} \in \bar{\alpha}^- \\ y_0 & \text{otherwise;} \end{cases}$$

(ii) For every $\sigma \in \Sigma$

$$H_{\sigma,new}(\mathbf{y}) = \begin{cases} H_\sigma(\mathbf{y}) & \text{if } \mathbf{y} \notin \bar{\alpha}^- \\ H_\sigma(\mathbf{y}) + E_\sigma(\mathbf{x})H_\alpha(\mathbf{y}) & \text{if } \mathbf{y} \in \bar{\alpha}^-; \end{cases}$$

Lemma 1

$$(\forall l \in \Sigma_u^*) l \in L(\mathbf{G}) \Rightarrow l \in L(\mathbf{E})$$

if and only if

$$(\forall l \in \Sigma_u^*) l \in L(\mathbf{G}) \Rightarrow l \in L(\mathbf{E}_{new}).$$

□

Lemma 2 Let $s \in L(\mathbf{E}_{new}) \cap L(\mathbf{E})$. Then

$$(\forall l \in \Sigma_u^*) sl \in L(\mathbf{G}) \Rightarrow sl \in L(\mathbf{E})$$

if and only if

$$(\forall l \in \Sigma_u^*) sl \in L(\mathbf{G}) \Rightarrow sl \in L(\mathbf{E}_{new}).$$

□

Theorem 3

$$supC(\mathbf{E}_{new}) = supC(\mathbf{E}).$$

□

We call an event $\alpha (\in \Omega)$ the *source event* wrt Ω ($\subseteq \Sigma_u$) if

$$(\forall \mathbf{x} \in \alpha^+) \mathbf{x}^+ \cap \Omega = \emptyset, \text{ i.e. } (\alpha^+)^+ \cap \Omega = \emptyset.$$

A source event α has no event in Ω 'ahead of' α. From [3] we know there always exists at least one such α if $\mathbf{G}$ is loop-free wrt Ω. For $\Omega = \Sigma_u$ we choose a source event α_1 wrt Ω. Let $\Omega = \Sigma_u - \{\alpha_1\}$ and choose a source event α_2 wrt the new Ω, $\cdots$. We get a finite sequence

$$\alpha_1, \alpha_2, \cdots, \alpha_p.$$

Define

$$\mathbf{E}_1 := \mathbf{E}$$

$$\mathbf{E}_{k+1} := T_{\alpha_k}\mathbf{E}_k \qquad (\text{ k=1,2,...,p}).$$

We have

Lemma 3

T_{α_k} does not decrease the elements of H_{α_i} (i=1,...k).

□

After the transformation T_{α_1}, the displacement vector corresponding to α_1 satisfies

$$H_{\alpha_1} \geq 0.$$

From Lemma 3 it follows that the component of H_{α_1} will never decrease after transition T_{α_i} $(i \geq 1)$. So after

$$T_{\alpha_1},\ T_{\alpha_2}, \cdots, T_{\alpha_p},$$

all the components of H_α ($\alpha \in \Sigma_u$) are nonnegative, implying that $L(\mathbf{E}_p)$ is controllable. Now we have the following main result:

Theorem 4 The process of computing $\mathbf{E}_{new}$ will stop after a finite number of iterations. Let the final VDES be $\mathbf{E}^*$. Then

$$supC(\mathbf{E}) = supC(\mathbf{E}^*) = L(\mathbf{E}^* \|_c \mathbf{G}).$$

□

4 Example

We consider a manufacturing system consisting of two groups of machines and a buffer. The machines in Group 1 take material from a store and discharge the finished product into a buffer B, from which the machines in Group 2 take the product for further processing. The specifications are as follows:

(i) The buffer B never overflows or underflows;

(ii) If some machines in both groups are broken down and require repair, the machines in Group 2 have 'priority' over those of Group 1. The 'priority' means that we must ensure there are at least m_0 repairers (out of the total of m repairers), prepared for the service of those of Group 2.

Let Group i be described as:

$$\mathbf{G}_i = (\mathbf{Z}^4, \Sigma_i, \delta_i, X_{0i}).$$

Here $\Sigma_i = \{\alpha_i, \beta_i, \lambda_i, \gamma_i, \mu_i\}$, $\Sigma_{ci} = \{\alpha_i, \gamma_i\}$. The event labels are:

α_i– A machine in Group i starts work;
β_i– A macnine in Group i finishes work;
λ_i–A machine in Group i breaks down;
γ_i–A machine in Group i requires repair;
μ_i– A machine in Group i is repaired.

The Petri Net graph of $\mathbf{G}_i$ is shown in Fig.1.

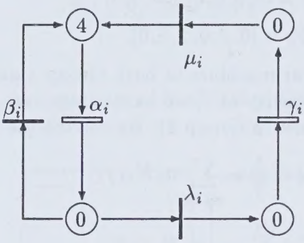


Fig.1 MACH i

Define $\mathbf{G}$=$\mathbf{G}_1||_c\mathbf{G}_2$. Then the displacement vectors of $\mathbf{G}$ are:

$$E_{\alpha_1} = [-1,1,0,0,0,0,0,0], \qquad E_{\alpha_2} = [0,0,0,0,-1,1,0,0],$$

$$E_{\beta_1} = [1,-1,0,0,0,0,0,0], \qquad E_{\beta_2} = [0,0,0,0,1,-1,0,0],$$

$$E_{\lambda_1} = [0,-1,1,0,0,0,0,0], \qquad E_{\lambda_2} = [0,0,0,0,0,-1,1,0],$$

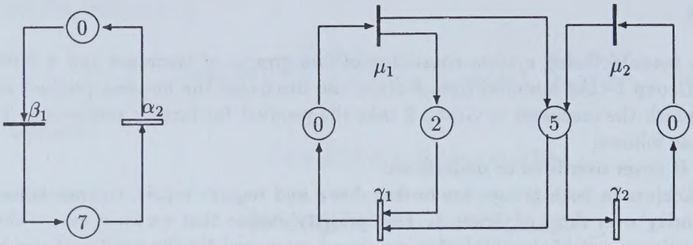
$$E_{\gamma_1} = [0,0,-1,1,0,0,0,0], \qquad E_{\gamma_2} = [0,0,0,0,0,0,-1,1],$$

$$E_{\mu_1} = [1,0,0,-1,0,0,0,0], \qquad E_{\mu_2} = [0,0,0,0,1,0,0,-1].$$

The specification is described as:

$$\mathbf{E} := SPEC1||_c SPEC2 = (Z^6, \Sigma_1 \cup \Sigma_2, \xi, Y_0),$$

as shown in Fig.2.



SPEC 1 SPEC 2

Fig.2 Specification

The displacement vectors are:

$$H_{\beta_1} = [-1, 1, 0, 0, 0, 0], \qquad H_{\alpha_2} = [1, -1, 0, 0, 0, 0],$$
$$H_{\mu_1} = [0, 0, -1, 1, 1, 0], \qquad H_{\mu_2} = [0, 0, 0, 0, 1, -1],$$
$$H_{\gamma_1} = [0, 0, 1, -1, -1, 0], \qquad H_{\gamma_2} = [0, 0, 0, 0, -1, 1].$$

Since **G** and **E** are loop-free wrt Σ_u, we can use the result of previous sections. For example, we begin from the initial state

$$X_0 = [4, 0, 0, 0, 4, 0, 0, 0]$$
$$Y_0 = [0, 7, 0, 2, 5, 0].$$

(This means that to start with, all four machines in both Group 1 and Group 2 are ready to work; the buffer B is full with the capacity of 7; we have 5 repairers, among them at least 3 repairers being used to service machines in Group 2). By solving the following 6 LIPs:

$$min(Y(j) + \sum_{\sigma \in \Sigma_u} n_\sigma H_\sigma(j))$$

$$s.t. \begin{cases} X + \sum_{\sigma \in \Sigma_u} n_\sigma E_\sigma \geq 0 \\ n_\sigma \geq 0, \qquad n_\sigma \ integer \end{cases}$$

for $Y = Y_0, X = X_0$, we know that all the solutions are nonnegative; this implies that

$$\epsilon \in supC(E).$$

If we know $s \in supC(\mathbf{E})$ take $X = \delta(s\alpha, X_0), Y = \xi(s\alpha, Y_0)$, in the former LIPs. If the solusions are all nonnegative then $\alpha \in \mathbf{V}^*(s)$; otherwise $\alpha \notin \mathbf{V}^*(s)$.

Furthermore, since **G** satisfies **A2**, we can get the closed form supervisory control of **G** as follows. From [3] we know that

$$\mathbf{S} = \mathbf{S}_1 \|_c \mathbf{S}_2,$$

with $\mathbf{S}_1 := supC(SPEC1), \mathbf{S}_2 := supC(SPEC2)$.

To construct $\mathbf{S}_1$, we first compute the source event sequence. In this case the source event sequence has only one element β_1. By doing the operation $T_{\beta_1}(SPEC1)$ we get $\mathbf{S}_1 = T_{\beta_1}(SPEC1)$ (Fig.3). We compute the new displacement vectors $H_{\sigma,new}$ as follows: (Notice that for VDES SPEC1 the displacement vectors are

$$H_{\beta_1} = [-1, 1],$$
$$H_{\alpha_2} = [1, -1],$$
$$H_\sigma = 0 \qquad \text{for } \sigma \notin \{\beta_1, \alpha_2\}.)$$

$$\mathbf{x}^{\Sigma} = \{\beta_1, \alpha_1, \lambda_1\}, \qquad \bar{\beta}_1^{-} = \{1\}.$$

$$\begin{aligned}
&H_{\beta_1,new}(1) = 0,\\
&H_{\beta_1,new}(2) = H_{\beta_1}(2) = 1,\\
&H_{\alpha_1,new}(1) = H_{\alpha_1}(1) + E_{\alpha_1}(2)H_{\beta_1}(1) = -1,\\
&H_{\alpha_1,new}(2) = H_{\alpha_1}(2) = 0.\\
&H_{\lambda_1,new}(1) = H_{\lambda_1}(1) + E_{\lambda_1}(2)H_{\beta_1}(1) = 1,\\
&H_{\lambda_1,new}(2) = H_{\lambda_1}(2) = 0.\\
&H_{\sigma,new} = H_{\sigma} \qquad \forall \sigma \notin \{\alpha_1, \beta_1, \lambda_1\}.
\end{aligned}$$

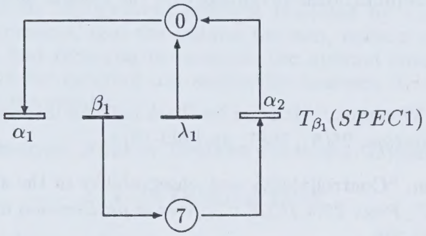

0
β_1
α_2
α_1
λ_1
$T_{\beta_1}(SPEC1)$
7

Fig.3 S$_1$=supC(SPEC1)

We can construct S$_2$ the same way. In this case the source event sequence is μ_1, μ_2. So S$_2 = T_{\mu_1}T_{\mu_2}(SPEC2)$. (Fig.4).

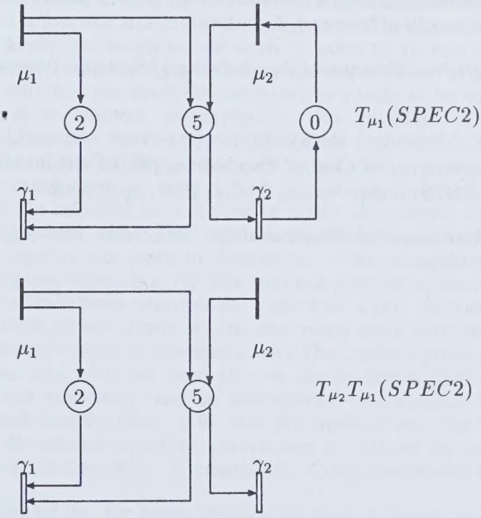

μ_1
μ_2
2
5
0
$T_{\mu_1}(SPEC2)$
γ_1
γ_2
μ_1
μ_2
2
5
$T_{\mu_2}T_{\mu_1}(SPEC2)$
γ_1
γ_2

Fig.4 S$_2$ =supC(SPEC2)

5 Conclusions

Most real-world discrete-event systems are complex both physically and computationally. To achieve computational efficiency is one of the motivations for studying VDES. In this paper, we discussed the supervisory control problem of VDES and provided algorithms for computing the optimal supervisory control with the specification given in the form of VDES. All the results are based on the condition of loop-freedom. If the plant **G** satisfies certain additional conditions we obtain a closed form solution to the problem.

The model we adopted here is based on the class of Petri-net languages L^f [5]. Neither of the two classes L^f, and the regular languages are included in the other. So we cannot say that VDES is the generalization of DES. But VDES is a very efficient model for both representation and computation, especially for multi-process problems. Because VDES is structured, we obtain a computational advantage over the general model of DES in such cases.

References

[1] P.J. Ramadge, W.M. Wonham. " Modular feedback logic for discrete event systems", *SIAM J. Control and Optimization*, 25(5), 1987, pp.1202-1218.

[2] Y. Li, W.M. Wonham. "Controllability and observability in the state feedback control of discrete event systems", *Proc. 27th IEEE Conference on Decision and Control*, Austin, TX, December 1988, pp.203-208.

[3] Y. Li, W.M. Wonham. " A state-variable approach to the modeling and control of discrete-event systems", *Proc. 26th Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, September 1988, pp.1140-1149.

[4] Y.Li, W.M. Wonham. "Linear integer programming techniques in the control of vector discrete-event systems", *Systems Control Group Report # 8909,* Department of Electrical Engineering, University of Toronto, June 1989.

[5] J.Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliff, N.J., 1981.

[6] W.M.Wonham. "A control theory for discrete-event systems", *in Advanced Computing Concepts and Techniques in Control Engineering (M.J.Denham and A.J.Laub,Eds.) NATO ASI Series* Vol F47, Springer-Verlag, Berlin, 1988, pp.129-169.

[7] J.F.Shapiro. *Mathematical Programming: Structures and Algorithms*, John Wiley & Sons,New York, 1979.

COMMENTS ON THE SENSITIVITY OF THE OPTIMAL COST AND THE OPTIMAL POLICY FOR A DISCRETE MARKOV DECISION PROCESS*

ENRIQUE L. SERNIK and STEVEN I. MARCUS**
**Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas 78712

ABSTRACT

The problem of characterizing the effects that uncertainties and/or small changes in the parameters of a model can have on optimal policies is considered. It is shown that changes in the optimal policy are very difficult to detect even for relatively simple models. By showing for a machine replacement problem modeled by a partially observed, finite state Markov decision process, that the infinite horizon, optimal discounted cost function is piecewise linear, we find formulas to compute the optimal cost and the optimal policy, thus providing a means for carrying out sensitivity analyses. Examples are presented to show the usefulness of the results.

Key words: Sensitivity analysis, Markov Decision Processes, Dynamic Programming.

1. INTRODUCTION

The problem of finding explicit descriptions and/or structural properties of optimal control laws and costs for partially observed (PO) stochastic control problems has received considerable attention in recent years (e.g., Ref. 1, 2). This is due in part to the computational advantages that have resulted from such descriptions (e.g., Ref. 3, 4, 5), and in part to the increasing interest in the design of adaptive control techniques aimed at overcoming changes in the optimal law caused by uncertainties and/or small changes in the parameters of the physical system being modeled (e.g., Ref. 6, 7, 8).

In this note we are intersted in finding how uncertainties and/or small changes in the parameters of the model affect the optimal policy and cost of a discrete PO Markov decision process (MDP). A simple example of such a process is that associated with a machine that produces items, namely a process that can be in either a "good" or in a "bad" state (corresponding to whether the machine produces or needs to be replaced). Since the state of the machine is monitored incompletely, this problem is converted to an equivalent completely observable (CO) MD problem (see e.g. Ref. 3, or Ref. 9, chapter 3), in which the conditional probability vector $\pi(t) = (\pi_1(t), \cdots, \pi_N(t))$, with $\pi_i(t)$ the probability that the machine is in state i ($i \varepsilon \{1, \cdots, N\}$) given past observations and actions, provides all the relevant information to select the control action at time t.

Even for this simple model, the effect that uncertainties on the parameters have on the optimal policy and cost is not easy to determine. The complication arises due to several reasons, including the following: (i) The optimal control action has to be specified for each of the (uncountably infinite number of) values of $\pi(t)$. This should be contrasted with the case of perfect observations where one need only compute the control for each of the finite number of values of the states; (ii) The control process for this production problem takes values in a finite set (namely one can let the machine produce, and either inspect or replace the machine), and so derivatives with respect to the control are not defined; (iii) It is well known (Ref. 1-9) that the optimal cost for the problem considered here satisfies a functional equation, which can be solved by using the Dynamic Programming (DP) algorithm (see Ref. 9, chapter 5). Computationally however, this is

---

* Research supported in part by the Air Force Office of Scientific Research under Grant AFOSR-86-0029, in part by the National Science Foundation under Grant ECS-8617860, and in part by the DoD Joint Services Electronics Program through the Air Force Office of Scientific Research (AFSC) Contract F49620-86-C-0045.

not an easy problem. The space in which $\pi(t)$ takes its values is discretized by means of a grid which has to be changed several times until one is certain that the result of (applying) the algorithm is actually a solution of the functional equation, and is not just an artificial result of the numerical discretization (see the examples below).

We investigate the dependence of the optimal cost and the optimal policy on any of the parameters of the model with two states before considering a more general model. The equations involved are scalar equations since $\pi(t)$ can be written as $\pi(t) = (1 - p(t), p(t))$, where $p(t)$ is the probability that the system is in the bad state at time t given past observations and actions; $p(t)$ will usually be denoted by p, omitting explicit dependence on t. The scalar formulation facilitates the determination of structural properties of the cost and the policy.

2. MODEL, NOTATION AND REVIEW OF PREVIOUS WORK

We denote by $\{x_t, t = 0, 1, \cdots\}$ the finite state MP associated with a machine that produces items; the state space is $X = \{0, 1\}$, also referred to henceforth as {good, bad}, respectively. Denote by $\{u_t, t = 0, 1, \cdots\}$ the control process; $u_t \varepsilon U = \{0, 1, 2\}$, also referred to as {produce, inspect, replace}, respectively. We associate a cost with each control action as follows: the cost associated with replacing the machine is denoted by R, and the cost associated with inspection by I. The cost associated with production is 0 if the machine is in the good state, and C if the machine is in the bad state. We assume that $0 < C < I < R$.

At time t one must decide whether to inspect the item produced or not, and whether to replace the machine or not. If the machine is replaced it will be in the good state at the end of period t. It is further assumed that no item was produced during that period. Inspection might be carried out to determine the state of the machine, but it will not imply a decision about whether to replace the machine. The observation process $\{y_t, t = 0, 1, \cdots\}$ takes values in $Y = \{0, 1\}$.

Assume that the probability vector $p^0 = (p_0^0, p_1^0)$ is given, where $p_i^0 = Pr\{x_0 = i\}$, $i = 0, 1$, and $\pi(0) = p^0$. The machine evolves according to transition probabilities $p_{ij}(u_t)$ defined by $p_{ij}(v) = Pr\{x_{t+1} = j / x_t = i, u_t = v\}$. Let $P(u_t)$, $u_t \varepsilon U$, be the transition probability matrices with entries $p_{ij}(u_t)$. The transition matrices $P(u_t)$, $u_t \varepsilon U$, are given by:

$$P(0) = P(1) = \begin{bmatrix} 1-\theta & \theta \\ 0 & 1 \end{bmatrix}, \quad P(2) = \begin{bmatrix} 1-\theta & \theta \\ 1-\theta & \theta \end{bmatrix}, \quad t = 0, 1, \cdots \tag{1}$$

where θ is the probability of machine failure in one time step.

The observation process is related to the state and the control processes by means of the conditional probabilities $q_{ik}(v) = P\{y_{t+1} = k / x_t = i, u_t = v\}$, with $q_{ik}(u_t)$ the entries of the observation matrices $Q(u_t)$, $u_t \varepsilon U$, given by:

$$Q(0) = Q(1) = Q(2) = \begin{bmatrix} q & 1-q \\ 1-q & q \end{bmatrix}, \quad t = 0, 1, \cdots \tag{2}$$

where $q \varepsilon [0.5, 1.0]$ is the probability of making a correct observation. The model is the one described by Ross in Ref. 10.

We are interested in the infinite horizon case, and the objective is to find an optimal admissible control policy that minimizes the expected discounted cost $J_g(p^0)$, given by:

$$J_g(p^0) = E_{p^0}\left[\sum_{t=0}^{\infty} \beta^t c(x_t, u_t) \right] \tag{3}$$

where $E_{p^0}[.]$ denotes conditional expectation with respect to p^0; β is the discount factor with $0 \leq \beta < 1$; $c(x_t, u_t)$ is the cost accrued when the machine is in state x_t and action u_t is selected; and $g = \{g_t\}_{t=1}^{\infty}$ is an admissible policy, that is, $\{g_t\}_{t=1}^{\infty}$ is a sequence of

Borel measurable maps $g_t : [0,1] \to U$ such that $u_t = g_t(p(t))$, $u_t \varepsilon U$, for $t = 0, 1, \cdots$. If no observations are available, u_t can still be written as $u_t = g_t(p(t))$, $u_t \varepsilon U$, and $g_t : [0,1] \to U$ for $t = 0, 1, \cdots$, where now $p(t)$ is the (aposteriori) probability that the machine is in state 1. This is because the expected cost can be expressed explicitly in terms of $p(t)$. However, in this case $\{g_t\}_{t=1}^{\infty}$ is a deterministic sequence since $p(t)$ depends only on p^0, which is given, and is updated from time t to time $t+1$ using the transition probabilities $P(.)$, also given. If $g_t(.) = g(.)$ for all values of t, the policy is said to be stationary (when computing optimal policies in the infinite horizon case, we need only consider stationary policies; see Ref. 9, p. 225).

Define $V_\beta(p) = \inf_g J_g(p)$. Then $V_\beta(p)$ is the expected cost accrued when an optimal policy is selected, given that the machine starts in the bad state with probability p, and future costs are discounted at rate β. It is well known (e.g. Ref. 9, 10) that $V_\beta(p)$ is the unique solution of:

$$V_\beta(p) = \min \{ C\,p + \beta \sum_{k=0}^{1} D(k,p,0)\, V_\beta(T(k,p,0)),$$
$$I + \beta \sum_{k=0}^{1} D(k,p,1)\, V_\beta(T(k,p,1)),$$
$$R + \beta \sum_{k=0}^{1} D(k,p,2)\, V_\beta(T(k,p,2)) \} \tag{4}$$

where $T(k,p,v)$ is the updated probability that the system is in state 1, given that k was observed and the control applied was v. $T(k,p,v)$ is given by $T(k,p,v) = N(k,p,v)/D(k,p,v)$, with $N(k,p,v) = (N_1(k,p,v), N_2(k,p,v))$, $D(k,p,v) = \sum_j N_j(k,p,v)$, and where $N_j(k,p,v)$ represents the probability that the next state is j given that the outcome is k and the control applied is v (see Ref. 9, 10 for details).

We review some previous work associated with the following two special cases of the strictly PO (i.e., partial observations during production and during inspection) problem: **Case A**: Only two actions are considered, namely $U = \{0, 2\}$, and the state of the system is not observed (i.e., the state is 'completely unobserved') during production, that is, all the entries in $Q(0)$ equal 0.5; **Case B**: Now $U = \{0, 1, 2\}$, and the state of the system is not observed during production, but perfect observations (i.e., the state is 'completely observed') are obtained during inspection, so that $Q(0)$ is as in **Case A** and $Q(1)$ is the 2-dimensional identity matrix. We recall that these two cases are of interest since the completely unobserved (CU) and the completely observed (CO) cases repectively provide upper and lower bounds for the optimal value of the cost in the strictly PO case (Ref. 11).

The first structural results associated with the optimal policy and cost for the models described above were given by Ross (Ref. 10). Among several results, Ross gave necessary and sufficient conditions for the stationary policy "produce for all values of p" to be optimal. Ross also showed that (i) every optimal policy produces for all $p \varepsilon [0, \theta]$; and that (ii) it is optimal to replace for values of p near 1. Other results by Ross included sufficient conditions to verify the existence of optimal policies. The conditions were stated in terms of the optimal cost and were thus hard to verify. The characterization of the stationary optimal policies was done by White (Ref. 3), who showed that among the stationary optimal policies there is a smaller class of optimal policies, called structured policies, such that one need only look for structured policies when solving equation (4). Wang's work (Ref. 12) was aimed at showing that a structured policy (called 'control-limit policy' when only two actions are considered) is optimal for the two action, CU case. Wang also gave analytic expressions for computing the optimal cost and the optimal policy for this problem. Although these results can be used to show that the stationary optimal cost is piecewise linear, Wang did not do so, and unfortunately his results have been referred to primarily as a "computational procedure" (see e.g., Ref. 13). Wang studied a more general model for the two action, CU case than the one being treated here, but he did not considered the case of three actions (closed loop). In a later work (Ref. 14), Wang generalized his results to the two action, CU N – dimensional ($N > 2$) case.

Let us point out that the previous results characterize the optimal policy and give some properties of the optimal cost function for the problems described above, but they do not give insight on what happens to the optimal policy or to the optimal cost if there is uncertainty in the knowledge of the parameters of the model, or if these parameters undergo (unexpected) small changes. Solving the problem again via Dynamic Programming may not be practical for the infinite horizon case in terms of computational effort, as will be illustrated in the examples below.

3. PIECEWISE LINEAR OPTIMAL COST

Motivated by our interest in determining the sensitivity of the optimal policy with respect to the parameters of the model, and since the study of the functional equation (4) provides little insight on how the optimal policy or the optimal cost change when the parameters of the model are subject to small changes, we focused our attention on the study of the DP (or successive approximations, Ref. 9, 10) algorithm used to solve equation (4). Specifically, we analyzed the algorithm given by:

$$V_\beta^1(p) = \min\{Cp\ ,\ I\ ,\ R\ \}$$

$$V_\beta^n(p) = \min\{Cp + \beta \sum_{k=0}^{1} D(k,p,0)\, V_\beta^{n-1}(T(k,p,0))\,,$$

$$I + \beta \sum_{k=0}^{1} D(k,p,1)\, V_\beta^{n-1}(T(k,p,1))\,,$$

$$R + \beta \sum_{k=0}^{1} D(k,p,2)\, V_\beta^{n-1}(T(k,p,2))\,\} \tag{5}$$

where n represents the iteration, and $V_\beta^n(p)$ is the minimal cost that can be obtained starting in state 1 with probability p and proceeding for n stages with costs discounted by a factor β. Because of the relative simplicity of algorithm (5), and since from the theory of contraction mappings it is guaranteed that algorithm (5) converges uniformly to the unique solution $V_\beta(p)$ of equation (4) as $n \to \infty$ (see e.g. Ref. 2), we were able to prove the piecewise linearity of the optimal cost function, and obtained analytic expressions to compute the optimal cost and the optimal policy. We show this next.

Consider first Case A described above. When $q = 0.5$ we have $D(k,p,0) = D(k,p,2) = ½$, $T(k,p,2) = \theta$, and $T(k,p,0)$ becomes (Ref. 9):

$$T(k,p,0) = p(1-\theta) + \theta \equiv Tp \tag{6}$$

Thus, Tp satisfies $Tp > p$ for $p \,\varepsilon\, [0,1)$, with unique fixed point $p = 1$. From algorithm (5), denote by $\bar{P}_k(p)$ the function generated by $\bar{P}_k(p) \equiv Cp + \beta \bar{P}_{k-1}(Tp)$, $k = 2, 3, \cdots$, with $\bar{P}_1(p) \equiv Cp$, and denote by $\bar{R}_k$ the function generated using $\bar{R}_k \equiv R + \beta \bar{P}_{k-1}(\theta)$, $k = 2, 3, \cdots$, with $\bar{R}_1 \equiv R$. By applying recursively (5) one obtains that:

$$\bar{P}_k(p) = Cp \sum_{i=0}^{k-1} \beta^i (1-\theta)^i + C \sum_{i=1}^{k-1} \beta^i (1-(1-\theta)^i) \tag{7}$$

$$\bar{R}_k = R + C \sum_{i=1}^{k-1} \beta^i (1-(1-\theta)^i) \tag{8}$$

Since $C < R$ by hypothesis, and $p < 1$, the first iteration of (5) gives the policy "produce for all $p \,\varepsilon\, [0,1]$". As k increases, if $\bar{P}_k(p) \neq \bar{R}_k$ for all $k \,\varepsilon\, N$, and all $p \,\varepsilon\, [0,1]$, one obtains Ross' result mentioned above, namely, that "produce for all $p \,\varepsilon\, [0,1]$" is the stationary, infinite horizon optimal policy. If on the contrary $\bar{P}_k(p) = \bar{R}_k$ for some $k \varepsilon N$ and for some $p < 1$, call it α_k, then the optimal cost at iteration k will be specified by:

$$V_\beta^k(p)=\begin{cases} C p \sum_{i=0}^{k-1} \beta^i (1-\theta)^i + C \sum_{i=1}^{k-1} \beta^i (1-(1-\theta)^i) & p \varepsilon [0,\alpha_k) \\ R + C \sum_{i=1}^{k-1} \beta^i (1-(1-\theta)^i) & p \varepsilon [\alpha_k, 1] \end{cases} \tag{9}$$

with optimal policy "produce for $p \varepsilon [0,\alpha_k)$ and replace for $p \varepsilon [\alpha_k, 1]$". The point we want to make is the following: for $n = k+1$ (and similarily for subsequent iterations) in order to compute $V_\beta^{k+1}(p)$ we need to perform a minimization in an interval of the form $[0,b)$, $0 < b < 1$, requiring the evaluation of $V_\beta^k(Tp)$, $p \varepsilon [0,b)$. But since $Tp > p$ for $p < 1$, the result of the minimization only specifies the cost function in an interval of the form $[0, T^{-1}b)$ at iteration $k+1$. This implies the following:
(i) Since $p = 1$ is the unique fixed point of $T^{-1}p = (p-\theta)/(1-\theta)$, and $T^{-1}p$ continues to decrease the size of the interval of the form $[0,b)$ as $k \to \infty$, there is an iteration, call it l, for which this interval is smaller than $[0,\theta)$ (say $[0,\gamma), \gamma < \theta$). Since $T^{-1}p < 0$ for $p \varepsilon [0,\gamma)$, the cost specified in $[0,\gamma)$ in iteration l will not enter into the computation of $V_\beta^{l+1}(p)$, $V_\beta^{l+2}(p)$, $\cdots$.
(ii) To specify the cost function on the remainder of the interval, namely $[T^{-1}\alpha_k, 1]$, algorithm (5) requires a minimization of the form:

$$\min \{Cp + \beta \bar{R}_k , \bar{R}_{k+1}\} \tag{10}$$

Note that in (10) we are comparing a function associated with the produce action, given by an affine function of p (referred to henceforth simply as a 'line segment' in order to facilitate the exposition), and $\bar{R}_{k+1}$, a constant. If $Cp + \beta \bar{R}_k$ is smaller than $\bar{R}_{k+1}$ in (10) (for some value of $p \varepsilon [T^{-1}\alpha_k, 1]$), a new line segment will appear in the description of the cost function (for the interval $p \varepsilon [T^{-1}\alpha_k, \alpha_{k+1})$). The point here is that with the exception of the line segment shown in (9), all the line segments appearing in the description of the cost function come from a minimization of the form (10).

In other words, from (ii), at each iteration of (5), and independently of the number of line segments already describing the cost function in that iteration, there is at most one new line segment appearing in the description of the optimal cost function, and from (i), for k large enough the line segments are also leaving the problem (meaning they no longer appear in the cost function) under the action of $T^{-1}p$.
(iii) In addition, observe that the line segment that specifies the cost function at iteration (say) k in the interval $[a,b)$, $0 < a < b < 1$, specifies (with formula updated by the iterative procedure) the cost function at iteration $k+1$ in the interval $[T^{-1}a, T^{-1}b)$, and since for $0 < a < b < 1$ we have that $T^{-1}b - T^{-1}a = (b-a)/(1-\theta) > b-a$, the line segments leaving the problem have finite nonzero length.

Finally, if we call α^* the limit as $k \to \infty$ of α_k (whenever it exists), then there is a finite natural number m_{α^*} such that $T^{-m_{\alpha^*}}\alpha^*$ is less than zero (this is clear because $T^{-1}p < p$ for $0 \le p < 1$, and the fact that $p = 1$ is the only fixed point of $T^{-1}p$). The same is true for each of the α_k, $k \varepsilon N$. This observation, together with remarks (i), (ii) and (iii) above, means that all the line segments that appear in the description of the cost function disappear in a finite number of iterations.

We introduce the following notation. Let $W_\beta^k(p) \equiv V_\beta^k(p)|_{[0,\alpha_k)}$, that is, $W_\beta^k(p)$ denotes the restriction of the optimal cost function at iteration k to the interval $[0,\alpha_k)$. Observe that in this notation $\bar{R}_k$ can be interpreted as the restriction of $V_\beta^k(p)$ to the interval $[\alpha_k, 1]$. Let $W_\beta(p)$ and $\bar{R}$ be the limits (whenever they exist) of $W_\beta^k(p)$ and $\bar{R}_k$ respectively, as $k \to \infty$. Then $W_\beta(p)$ (respectively $\bar{R}$) denotes the restriction of the infinite horizon, optimal discounted cost function $V_\beta(p)$ to the interval $[0,\alpha^*)$ (respectively $[\alpha^*, 1]$).

One of two things can happen so that convergence is achieved: either **(a)** The line segments enter into the problem at a higher rate than that at which they disappear from the problem, and hence the line segments accumulate, meaning that in the limit as $k \to \infty$, $W_\beta(p)$ will not be described by a finite number of line segments; or else **(b)** The rate at which the line segments enter the problem is the same as that at which they leave the problem, and hence a finite number of line segments completely describe $W_\beta(p)$. We have the following Proposition.

Proposition 3.1: For the open loop model described above in *Case A* (two actions, i.e., $U = \{\text{produce}, \text{replace}\}$, and the state of the system is CU during production), $W_\beta(p)$, associated with the stationary, infinite horizon optimal policy is piecewise linear. Since $\bar{R}$ is constant, this means that the infinite horizon, optimal cost function $V_\beta(p)$ is piecewise linear.

Proof: By Lemma 3.2 in Ref. 10, we have that every optimal policy produces for all $p \varepsilon [0,\theta]$. Hence we consider the following two cases:
(i) If $\alpha^* = \theta$, then $T^{-1}\alpha^* = 0$, and the claim here is that the optimal cost $V_\beta(p)$ associated with the stationary, infinite horizon optimal policy "produce for $p \varepsilon [0,\theta]$ and replace for $p \varepsilon (\theta, 1]$", is given by:

$$V_\beta(p) = \begin{cases} Cp + \beta\bar{R} & p \varepsilon [0,\theta] \\ \bar{R} & p \varepsilon (\theta, 1] \end{cases} \tag{11}$$

For if not, assume that $W_\beta(p)$ is an arbitrary limit of piecewise linear functions. Denote by $f(p)$ the optimal cost function that is an arbitrary limit of piecewise linear functions for $p \varepsilon [0,\alpha^*]$, and a constant for $p \varepsilon (\alpha^*, 1]$. Then from (4) we have that: $f(p) = \min\ \{Cp + \beta f(Tp), R + \beta f(\theta)\}$. That is, by Lemma 3.2 in Ref. 10, and for $0 \le p \le \alpha^* = \theta$, we have $f(p) = Cp + \beta f(Tp)$. Since $f(Tp) = R + \beta f(\theta)$ is constant (say K) for $0 \le p \le \alpha^* = \theta$, we have $f(p) = Cp + K$ for $p \varepsilon [0, \alpha^*]$. Therefore, if $\alpha^* = \theta$, the infinite horizon, optimal cost is given by (11), which means that there is only one line segment describing $W_\beta(p)$.
(ii) Since the case for which $\alpha^* = 1$ was considered by Ross in Ref. 10, assume that $\theta < \alpha^* < 1$, and that convergence takes place as described in **(a)** above. As explained in remark (ii) above, the line segment that appeared in the problem most recently specifies the cost function in the interval $[T^{-1}\alpha^*, \alpha^*)$. Since $\alpha^* > T^{-1}\alpha^*$, this line segment has finite nonzero length. Now assume that for $p \varepsilon [0, T^{-1}\alpha^*)$ the cost function is arbitrary. Denote by $f(p)$ the optimal cost function that is an arbitrary limit of piecewise linear functions for $p \varepsilon [0, T^{-1}\alpha^*)$, an affine function of p for $p \varepsilon [T^{-1}\alpha^*, \alpha^*)$, and constant for $p \varepsilon [\alpha^*, 1]$. From (4), $f(p)$ is given by $f(p) = \min\ \{Cp + \beta f(Tp), R + \beta f(\theta)\}$. Since for $p \varepsilon [T^{-1}(T^{-1}\alpha^*), T^{-1}\alpha^*)$ the optimal policy is to produce, $f(p) = Cp + \beta f(Tp)$ for $p \varepsilon [T^{-1}(T^{-1}\alpha^*), T^{-1}\alpha^*)$. But for $p \varepsilon [T^{-1}(T^{-1}\alpha^*), T^{-1}\alpha^*)$ we have $Tp \varepsilon [T^{-1}\alpha^*, \alpha^*)$, and so $f(Tp)$ is the above mentioned segment in $[T^{-1}\alpha^*, \alpha^*)$ with finite (nonzero) length, which in turn means that $f(p) = Cp + \beta f(Tp)$ is also an affine function of p, and since $T^{-1}\alpha^* > T^{-1}(T^{-1}\alpha^*)$, it also has a finite (nonzero) length. By remark (iii) above, we have that $\alpha^* - T^{-1}\alpha^* > T^{-1}\alpha^* - T^{-1}(T^{-1}\alpha^*)$. Therefore, continuing the procedure just described, we can see that there is a uniform lower bound (different from zero) for the length of all the line segments which is independent of the iteration because $T^{-1}p$ is independent of the iteration. Taking into account the previous observations, we conclude that this uniform lower bound for the length of the line segments implies an upper bound in the number of line segments describing $V_\beta(p)$. **QED**

At this point the following remarks are in order:
(i) To the best of our knowledge, the piecewise linearity of the optimal cost function in the infinite horizon model for the cases described above has not been reported previously (see for example all the references cited so far, and recent reviews like Ref. 16). Its usefulness will be apparent in the sequel.

(ii) Although the two action, CU model may be of limited interest in practice (namely, it may only be used in some replacement models in which the equipment is subject to breakdowns but no measurements are available, and therefore only open loop control is applicable), its importance here resides in that the insight obtained by its study allowed us to perform a similar analysis (and also prove the piecewise linearity of the optimal cost function) for the three action, closed loop case.
(iii) The analysis of the successive approximations algorithm for the case of three actions is not trivial. As pointed out in Ref. 15, p. 29, a policy can appear at any time during the iterative procedure, yet fail to be optimal for the infinite horizon case. Furthermore, it might also happen that a policy appears during some iteration, does not appear in the (e.g., one hundred) subsequent iterations, and reappears later (or never reappears), meaning that a policy structure which is not any longer optimal after some finite iteration $k \varepsilon \boldsymbol{N}$, cannot be eliminated as suboptimal, and so, estimation of the minimum number of iterations required (for example in algorithm (5)) to guarantee that an optimal policy for the n^{th} (finite) horizon is also the optimal policy for the infinite horizon case, remains an open problem (Ref. 15).

For the sake of brevity, we will not go into the details of the three action, closed loop problem, since it requires a lenghty analysis of the successive approximations algorithm (5). However, we have studied the algorithm (5) and shown that some policy structures cannot occur at all during the iterative procedure (e.g., there is not a "produce-inspect" policy), and that the occurrence of others do not affect the stationary, infinite horizon policy structure (Ref. 17).

With these results we were able to show that the infinite horizon, optimal cost function in the three action, closed loop problem is piecewise linear, and were able to develop analytic expressions (equivalent to those of Wang in Ref. 12 for the two action, CU case, and new ones for the three action, closed loop case) for the costs and the structured optimal policies for **Cases A** and **B** (namely, policies that as a function of p have the characterization: "there exist three numbers ρ_i, $i = 1,2,3$, $\theta \leq \rho_1 \leq \rho_2 \leq \rho_3 \leq 1$, such that it is optimal to produce for $0 \leq p < \rho_1$ ($0 \leq p \leq \rho_1$ if $\rho_1 = \theta$) and $\rho_2 \leq p < \rho_3$, it is optimal to inspect for $\rho_1 \leq p < \rho_2$, and it is optimal to replace for $\rho_3 \leq p \leq 1$"; see Ref. 3, 10 for detailed analysis). This in turn allows us to avoid the computational burden described in Section 2 when solving the control problem: in particular, the results allow us to perform a sensitivity analysis of the optimal policy with respect to any of the parameters of the problem, as will be illustrated in the examples ahead.

Once it is established that only a finite number of line segments is required to completely describe the infinite horizon optimal cost function, algorithm (5) can be used to find analytic expressions to compute the cost and the policy structure. Since the importance of these formulas reside in their use to perform sensitivity analyses of the optimal cost and policy, we illustrate this next with some examples.

4. EXAMPLES

Example 1: Consider the closed loop problem with the following data: $\beta = 0.985$, $\theta = 0.1$, $C = 4.0$, $I = 5.56$ and $R = 10.0$. The stationary, infinite horizon optimal policy is: "produce for $p \varepsilon [0, 0.5145193)$ and $p \varepsilon [0.5369462, 0.6070793)$, inspect for $p \varepsilon [0.5145193, 0.5369462)$ and replace for $p \varepsilon [0.6070793, 1]$", and the associated optimal cost is given by:

$$
V_{\beta}(p) = \begin{cases}
23.33618p + 151.88782 & p\,\varepsilon\,[0.0000000, 0.0864824) \\
21.81182p + 152.01965 & p\,\varepsilon\,[0.0864824, 0.1778342) \\
20.09230p + 152.32544 & p\,\varepsilon\,[0.1778342, 0.2600507) \\
18.15262p + 152.82985 & p\,\varepsilon\,[0.2600507, 0.3340457) \\
15.96460p + 153.56075 & p\,\varepsilon\,[0.3340457, 0.4006411) \\
13.49645p + 154.54959 & p\,\varepsilon\,[0.4006411, 0.4605770) \\
10.71230p + 155.83191 & p\,\varepsilon\,[0.4605770, 0.5145193) \\
7.57168p + 157.44782 & p\,\varepsilon\,[0.5145193, 0.5369462) \\
7.54600p + 157.46161 & p\,\varepsilon\,[0.5369462, 0.5634215) \\
4.00000p + 159.45950 & p\,\varepsilon\,[0.5634215, 0.6070793) \\
161.88782 & p\,\varepsilon\,[0.6070793, 1.0000000]
\end{cases} \tag{12}
$$

Observe that equation (12) is a closed form formula for the infinite horizon discounted cost. Also, note that there are 7 line segments describing the optimal cost function for $p\,\varepsilon\,[0.0, 0.5145193)$, and 2 line segments describing the optimal cost function for $p\,\varepsilon\,[0.5369462, 0.6070793)$. Now change θ from 0.1 to 0.10005. The optimal policy structure now changes from 4 to 2 regions. In this case $\alpha^* = 0.60721$, and the optimal cost is given by:

$$
V_{\beta}(p) = \begin{cases}
23.3210p + 151.9233 & p\,\varepsilon\,[0.00000, 0.08713) \\
21.7959p + 152.0562 & p\,\varepsilon\,[0.08713, 0.17846) \\
20.0754p + 152.3632 & p\,\varepsilon\,[0.17846, 0.26065) \\
18.1346p + 152.8691 & p\,\varepsilon\,[0.26065, 0.33463) \\
15.9452p + 153.6017 & p\,\varepsilon\,[0.33463, 0.40120) \\
13.4753p + 154.5927 & p\,\varepsilon\,[0.40120, 0.46111) \\
10.6890p + 155.8774 & p\,\varepsilon\,[0.46111, 0.51502) \\
7.5458p + 157.4962 & p\,\varepsilon\,[0.51502, 0.56355) \\
4.0000p + 159.4945 & p\,\varepsilon\,[0.56355, 0.60721) \\
161.9233 & p\,\varepsilon\,[0.60721, 1.00000]
\end{cases} \tag{13}
$$

Note how a relatively small change in the value of θ resulted in a significant change in the optimal policy structure. To the best of our knowledge, changes in the optimal policy due to such small changes on the parameters of the model could not be studied before, because the necessity of discretization often does not permit high confidence in the results obtained by following the DP algorithm. We are able to study small changes in the parameters of the model because of the analytic expressions found as a consequence of the piecewise linearity of the optimal cost.

Now consider the case when θ changes to $\theta = 0.09$. The optimal policy is: produce for $p\,\varepsilon\,[0, 0.4874221)$ and $p\,\varepsilon\,[0.5772912, 0.5773791)$, inspect for $p\,\varepsilon\,[0.4874221, 0.5772912)$, replace for $p\,\varepsilon\,[0.5773791, 1]$, and the optimal cost is given by:

$$V_\beta(p) = \begin{cases} 25.71507p + 143.96777 & p\,\varepsilon\,[0.0000000\,,0.0080949\,) \\ 24.22611p + 143.97982 & p\,\varepsilon\,[0.0080949\,,0.0973664\,) \\ 22.56497p + 144.14156 & p\,\varepsilon\,[0.0973664\,,0.1786034\,) \\ 20.71174p + 144.47256 & p\,\varepsilon\,[0.1786034\,,0.2525291\,) \\ 18.64421p + 144.99467 & p\,\varepsilon\,[0.2525291\,,0.3198015\,) \\ 16.33760p + 145.73232 & p\,\varepsilon\,[0.3198015\,,0.3810193\,) \\ 13.76427p + 146.71281 & p\,\varepsilon\,[0.3810193\,,0.4367276\,) \\ 10.89336p + 147.96661 & p\,\varepsilon\,[0.4367276\,,0.4874221\,) \\ 7.69048p + 149.52777 & p\,\varepsilon\,[0.4874221\,,0.5772912\,) \\ 4.00000p + 151.65825 & p\,\varepsilon\,[0.5772912\,,0.5773791\,) \\ 153.96777 & p\,\varepsilon\,[0.5773791\,,1.0000000\,] \end{cases} \qquad (14)$$

The optimal policy for this example still has four regions if θ changes to any value in $[0.09\,,0.10)$. However, note that the number of line segments describing the optimal cost function changes for different values of θ. Also, note the size of the interval for which the line segment with formula $4.0p + 151.65825$ is specified in (23). We compared these results with those given by the successive approximations algorithm. The results were very difficult to obtain by using the successive approximations algorithm. Unless one knows in advance the structure of the stationary, infinite horizon optimal policy, it is very difficult to decide when the optimal policy has been reached (and hence to decide when to stop the computational procedure), even after several choices of the grid have been tested (with the corresponding time consumption involved).

It is clear that the same kind of analysis carried out here can be done for any of the other parameters of the problem (i.e., β, R, I and C). Thus the equations found in this work can be used to obtain insight in the way the system responds to uncertainties, and therefore adaptive policies can be designed (for example, to modify the value of some of the parameters to compensate for an undesired change in some other parameters) so that the system continues to perform in a preselected satisfactory way.

Example 2: Now consider the open loop case. Let $\beta = 0.9999$, $\theta = 0.1$, $C = 4.0$ and $R = 10.0$. Using algorithm (5) with a grid of 1001 points in the interval $[0,1]$, one obtains: for $n = 1000$, $\alpha^* = 0.609$, $\bar{R} = 2272.10$; for $n = 10000$, $\alpha^* = 0.588$, $\bar{R} = 15077.17$; and for $n = 15000$, $\alpha^* = 0.601$, $\bar{R} = 18528.28$. We note that the last case mentioned above took 52 minutes of CPU (as compared to less than a second when using the expressions derived here, for the same computer and computer load), and although the values obtained may suffice when solving the (initial) control problem (actual values, obtained with the analytical expressions, are $\bar{R} = 23882.62$ and $\alpha^* = 0.597167$) it is apparent that a sensitivity analysis would be not only expensive in terms of computer time (this is so for any computational algorithm since p takes uncountably many values, and we are dealing with the infinite horizon problem), but also hard to perform in the sense of detecting the actual effect of the uncertainties on the optimal cost and policy.

5. CONCLUSIONS

The analysis of the successive approximations algorithm used to solve the functional equation satisfied by the optimal cost associated with the problems described in Section 3, allowed us to prove the piecewise linearity of the optimal cost function, and to develop analytical expressions to compute both the infinite horizon optimal cost and the stationary, infinite horizon optimal policy structure. These results, in turn, permit the performance of a sensitivity analysis for the optimal cost and policy with respect to any of the parameters of the problem.

The examples in Section 4 suggest that for the study of changes in the optimal policy due to small changes in the parameters of the model, better results can be obtained if structural properties of the policies and the cost are taken into account in the design of computational procedures. Since the strictly PO case is difficult to treat analytically, the study of structural properties of the optimal cost and the optimal policy for special cases of the strictly PO case, like those considered here, is justified as a way to approach the strictly PO problem.

The development of similar results for more complex problems like the study of the strictly PO case, higher dimensional models, and the average cost case, is currently being investigated.

6. REFERENCES

[1] Hopp, W. J., *Sensitivity Analysis in Discrete Dynamic Programming*, Journal of Optimization Theory and Applications, Vol. 56, No. 2, February 1988, p. 257-269.

[2] Ohnishi, M., Kawai, H. and Mine, H., *An Optimal Inspection* and *Replacement Policy Under Incomplete State Information*, European Journal of Operatonal Research, Vol. 27, 1986, p. 117-128.

[3] White, C. C., *Optimal Inspection* and *Repair of a Production Process Subject to Deterioration*, Journal of the Operational Research Society, Vol. 29, No. 3, 1978, p. 235-243.

[4] White, C. C., and El-Deib, H. K., *Parameter Imprecision in Finite State, Finite Action Dynamic Programs*, Operations Research, Vol. 34, No. 1, Jan/Feb 1986, p. 120-129.

[5] White, C. C., Thomas, L. C., and Scherer, W. T., *Reward Revision for Discounted Markov Decision Problems*, Operations Research, Vol. 36, No. 6, Nov/Dec 1985, p. 1299-1315.

[6] Hernandez-Lerma, O., and Marcus, S. I., *Adaptive Policies for Discrete-Time Stochastic Control Systems with Unknown Disturbance Distributions*, Systems & Control Letters, Vol. 9, 1987, p. 307-325.

[7] Hernandez-Lerma, O., and Marcus, S. I., *Adaptive Control of Discrete Discounted Markov Decision Chains*, Journal of Optimization Theory and Applications, Vol. 46, No. 2, June 1985, p. 227-235.

[8] Hernandez-Lerma, O., and Marcus, S. I., *Adaptive Control of Markov Processes with Incomplete State Information and Unknown Parameters*, Journal of Optimization Theory and Applications, Vol. 52, No. 2, February 1987, p. 227-241.

[9] Bertsekas, D. P., *Dynamic Programming*, Prentice Hall, Englewood Cliffs, New Jersey, 1987.

[10] Ross, S. M., *Quality Control Under Markovian Deterioration*, Management Science, Vol. 17, No. 9, 1971, p. 587-596.

[11] Astrom, K. J., *Optimal Control of Markov Processes with Incomplete State Information*, Journal of Mathematical Analysis and Applications, Vol 10, 1965, p. 174-205.

[12] Wang, R. C., *Computing Optimal Control Policies - Two Actions*, Journal of Applied Probability, Vol. 13, 1976, p. 826-832.

[13] Monahan, G. E., *A Survey of Partially Observable Markov Decision Processes: Theory, Models and Algorithms*, Management Science, Vol. 28, No. 1, Jan. 1982, p. 1-16.

[14] Wang, R. C., *Optimal Replacement Policy with Unobservable States*, Journal of Applied Probability, Vol. 14, 1977, p. 340-348.

[15] Federgruen, A., and Schweitzer, P. J., *Discounted and Undiscounted Value-Iteration in Markov Decision Problems: A Survey*, in Dynamic Programming and its Applications, edited by M. Puterman, Academic Press, 1979, p. 23-52.

[16] White, C. C., and White, D. J., *Markov Decision Processes*, European Journal of Operational Research, Vol. 39, 1989, p. 1-16.

[17] Sernik, E. L., and Marcus, S. I., *Sensitivity of the Optimal Cost and Policy for a Discrete Markov Decision Process*, submitted for publication.

OBJECT-ORIENTED COMPUTATION AND SIMULATION OF LARGE-SCALE DISCRETE-EVENT SYSTEMS

S.D. O'YOUNG AND W. M. WONHAM

Systems Control Group, Department of Electrical Engineering
University of Toronto, Toronto, Ontario, Canada M5S 1A4

ABSTRACT *We create a class of discrete-event system objects with the class name DES to encapsulate the transition structures of finite state machines. From this generic class DES, subclasses FSM (finite state machine), CP (Cartesian product), SP (synchronous product) and SH (shuffle product) are defined to model both horizontal and vertical architectures. Architectural objects are DES objects which are built on top of other DES objects. Thanks to this recursive definition of the architectural objects, a large-scale plant can be modeled modularly and consistently using DES objects. The modularity of this object-oriented approach also makes it well suited for parallel and distributed processing, an essential requirement for handling large-scale discrete event system computation and simulation.*

1.0 INTRODUCTION

Using the object-oriented methodology, we create a class of discrete-event system objects with the generic class name DES to encapsulate the transition structures of finite state machines at various levels of architectural abstraction in the modeling of a large-scale discrete-event system. This approach is entirely modular and decentralized in the sense that each instance of the class DES can be regarded an independent executable process which communicates with other DES objects via message-passing. A divide-and-conquer strategy can be implemented if the computation is done in a parallel and distributed environment.

The generic class DES contains a kernel set of class methods which enables an instance of its class or subclasses to recognize a string in the closed and marked languages generated by its encapsulated transition structure. From this generic class DES, subclasses FSM, CP, SP and

SH are defined to model 'machines' or groups of 'machines' within a large-scale discrete-event plant. Subclass FSM is a template for instantiating an object which encapsulates locally the entire (flat) transition structure of an automaton. Subclasses CP, SP and SH contain templates for instantiating 'architectural' objects representing Cartesian product, synchronous product and shuffle product transition structures respectively. Architectural objects are DES objects which are built on top of other child DES objects. Objects instantiated from these product subclasses do not contain local databases; instead they obtain all the information needed to construct their own transition structures from their children. The children can either be low-level flat FSM objects or high-level architectural objects such as other instances of the product subclasses. Thanks to the recursive nature of the definition of the product subclasses, a large-scale plant of arbitrary complexity can be modeled modularly and consistently using DES objects.

In the next section, we will describe a kernel set of methods which a DES object needs in order to recognize the closed and marked languages generated by its encapsulated transition structure. This kernel set of methods forms the class methods for the generic class DES. The subclasses of DES: CP, SP and SH are then defined in Section 3 by refining the methods within the kernel set to suit their specific synchronization requirements. The modeling and computing issues pertaining to these DES objects are discussed in Section 4.

2.0 SPECIFICATIONS FOR THE CLASS DES

2.1 Introduction

We begin by reviewing the definitions of a finite state machine within the Ramadge-Wonham framework [1]. A finite state machine is a fundamental building block for constructing discrete-event system computation and simulation architectural objects. A kernel set of methods necessary to construct a recognizer for the closed and marked languages generated by a finite state machine is then specified, and a generic class DES of discrete-event system objects is in turn defined by incorporating this kernel set as its class methods.

2.2 Finite State Machines

In the Ramadge-Wonham framework, a discrete-event system is represented by its state description or transition structure. Consider the case of a 'simple' machine with 3 states: *idle*, *working* and *down* as shown in Fig. 1, where the *idle* state is both the *initial* and *marker* (exit) state. This simple machine has an event label set $\Sigma = \{start,\ finish,\ fail,\ repair\}$ representing the transition triplets of the form [*exit state*, *event label*, *entrance state*]. For example, this simple machine has only one transition [*idle*, *start*, *working*] exiting from the initial state *idle* and entering the next state *working*. A control structure can be imposed upon the event labels by associating with them attributes like *controllable* and *forcible* [2]. Hierarchial control structure can also be encoded within the states by attributing vocalization labels [3] to a subset of the states.

Formally, the transition structure of a discrete-event system can be described by an automaton

$$G = \{\Sigma,\ Q,\ \delta,\ q_0,\ Q_m\} \tag{2.1.1}$$

where Σ is the set of event labels, Q is the set of states, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marker states and $\delta : \Sigma \times Q \to Q$, the (partial) transition function.

Let Σ^* denote the set of all finite strings over Σ including the null string ϵ. Extend δ to act over Σ^* by defining, for all $s \in \Sigma^*$, and $\sigma \in \Sigma$

$$\delta(\epsilon,\ q) := q \quad \text{and} \quad \delta(s\sigma,\ q) := \delta(\sigma,\ \delta(s,\ q)).$$

Then the *closed language* generated by G is defined to be $L(G) = \{s \in \Sigma^* \mid \delta(s,\ q_0) \text{ is defined}\}$ and the *marked language* generated by G is $L_m(G) = \{s \in \Sigma^* \mid \delta(s,\ q_0) \in Q_m\}$.

2.3 Methods of the Generic Class DES

Suppose that we would like to construct a *recognizer*, a machine which would recognize a string s in the closed or marked languages generated by the automaton G as defined in Section 2.2. One approach is to simulate stepwise the transition function δ by resetting G to its initial state and test whether or not G can be driven sequentially from the first through to the last transition as labeled by the alphabet symbols in s. To implement this, we need the following set of methods or commands for interacting with G:

`reset`(G): resets G to its initial state; fails if $L(G)$ is empty;

`state`(G, S):	returns the present state S of G and returns a null state if $L(G)$ is empty.
`go_to`(G, S):	sets the present state of G to S; fails if $S \notin Q$, the state set of G;
`marker`(G, S):	returns true if state $S \in Q$ is marked; otherwise fails;
`ntran`(G, Ev, En):	returns an event label Ev and the entrance state En associated with σ exiting from the present state of G; fails if there is no transition exiting from the present state of G.

Note that the null state of an empty language is defined to be marked. And in the definition of `ntran`, we have included the possibility that the underlying automaton might be nondeterministic. There might be more than one transition sharing the same exit state and event label but entering different entrance states.

Definition *The set {* `reset`, `state`, `go_to`, `marker`, `ntran`*} is said to be the kernel set of discrete-event system methods.*

Definition *A class DES is a template for instantiating objects which can process the kernel set of discrete-event system methods.*

Hence an automaton G (2.1.1) can be *objectified* by creating an instance of Class DES which encapsulates within its internal database the transition structure represented by the set $\{\Sigma,\ Q,\ \delta,\ q_0,\ Q_m\}$. Equipped with the kernel set of discrete-event system methods, the object can reconstruct the closed and marked languages generated by the automaton. It is convenient to define a subclass of DES for the instantiation of DES objects which encapsulate the entire flat transition structures within their local databases.

Definition *The Class FSM is a subclass of DES and is a template for instantiating objects which can process the kernel set of discrete-event system methods locally.*

3.0 PRODUCT STRUCTURES

We extend the idea from Section 2.3 to objectify product structures by defining the kernel set of discrete-event system methods specific to their respective synchronization requirements. This objectification is possible because the product formed by n automata is again an automaton.

3.1 Boundary Conditions

Consider a generic product structure $mach$ with n children: $mach_1$, $mach_2$, $\cdots$, $mach_n$,

$$mach = product\ (mach_1,\ mach_2,\ \cdots,\ mach_n)$$

where *product* can be a Cartesian product, a synchronous product or a shuffle product [4]. We will deal with the boundary condition where the product structure generates an empty language.

The closed language generated by a Cartesian product $mach$ is empty if *one or more* of its constituent discrete-event systems generate an empty closed language. The closed language generated by a synchronous product or shuffle product is empty if the closed language generated by *each* of its constituent discrete-event systems is empty. Hence, the `reset` command for an objectified Cartesian product $mach$ is implemented as follows.

$$\begin{aligned} &\texttt{reset}\ (mach): \\ &\qquad \text{executes} \quad \texttt{reset}\ (mach_1),\ \texttt{reset}\ (mach_2),\ \cdots,\ \texttt{reset}\ (mach_n); \\ &\qquad\quad \text{fails} \quad \text{if } \exists\ k,\ 1 \le k \le n,\ \texttt{reset}\ (mach_k) \text{ fails.} \end{aligned} \tag{3.1.1}$$

For an objectified synchronous product or shuffle product $mach$, the `reset` command is implemented as

$$\begin{aligned} &\texttt{reset}\ (mach): \\ &\qquad \text{executes } \texttt{reset}\ (mach_1),\ \texttt{reset}\ (mach_2),\ \cdots,\ \texttt{reset}\ (mach_n); \\ &\qquad\quad \text{fails} \quad \text{if } \forall\ k,\ 1 \le k \le n,\ \texttt{reset}\ (mach_k) \text{ fails.} \end{aligned} \tag{3.1.2}$$

3.2 Product Structure Methods

A *state* of $mach$ is defined in terms of the n-tuple $[state_1,\ state_2,\ \cdots,\ state_n]$ where $state_k$ is a state of $mach_k$. The product structure $mach$ is at its initial state iff $mach_1$, $mach_2$, $\cdots$, $mach_n$ are all at their respective initial states, and $mach$ at a marker state if $mach_1$, $mach_2$, $\cdots$, $mach_n$ are all at their respective marker states.

For a generic product structure *mach*, command `state` can be implemented as follows.

$$\texttt{state}\ (mach,\ S): \tag{3.2.1}$$

$$\text{returns}\quad S = [state_1,\ state_2,\ \cdots,\ state_n],\ \text{if}$$

$$\forall\ i,\ 1 \leq i \leq n,\ \texttt{state}\ (mach_i,\ state_i)\ \text{is true;}$$

$$\text{fails}\quad \text{otherwise.}$$

Command `go_to` can be implemented as follows.

$$\texttt{go\_to}\ (mach,\ [state_1,\ state_2,\ \cdots,\ state_n]): \tag{3.2.2}$$

$$\text{executes}\quad \forall\ i,\ 1 \leq i \leq n,\ \texttt{go\_to}\ (mach_i,\ state_i);$$

$$\text{fails}\quad \text{if}\ \exists\ k,\ 1 \leq k \leq n,\ \texttt{go\_to}\ (mach_k,\ state_k)\ \text{fails.}$$

Finally, command `marker` can be implemented as follows.

$$\texttt{marker}\ (mach,\ [state_1,\ state_2,\ \cdots,\ state_n]): \tag{3.2.3}$$

$$\text{returns}\quad \text{true if}\ \forall\ i,\ 1 \leq i \leq n,\ \texttt{marker}\ (mach_i,\ state_i)\ \text{is true;}$$

$$\text{fails}\quad \text{otherwise.}$$

In order to complete our object-oriented representation of the product structures, we need to specify the remaining command `ntran` which is defined according to the synchronization rules governing the respective product structures.

3.3 Cartesian Products

For the Cartesian product structure, all the next event labels must be synchronized. Hence, we have

$$\texttt{ntran}\ (mach,\ Ev,\ En): \tag{3.3.1}$$

$$\text{returns}\quad \text{event label}\ Ev = \sigma\ \ \text{and}\ En = [state_1,\ state_2,\ \cdots,\ state_n]$$

$$\text{if}\ \forall\ i,\ 1 \leq i \leq n,\ \texttt{ntran}\ (mach_i,\ \sigma,\ state_i)\ \text{is true;}$$

$$\text{fails}\quad \text{otherwise.}$$

3.4 Synchronous Products

For the synchronous product structure, all the child discrete-event systems must synchronize all the shared event labels. For the definition of `ntran` for the subclass synchronous product, it is convenient to define a command `next_event` as sub-command of `ntran` as follows

`next_event`(G, Ev): returns $Ev = \sigma$ if `ntran`(G, σ, _) is true; otherwise fails,

where the underscore "_" is taken to mean that any entrance state is acceptable. The command `next_event` simply means that one is interested only in the next transition label and is indifferent to the entrance state(s) corresponding to the transition label.

Let Σ_1, Σ_2, $\cdots$, Σ_n be the event label sets from $mach_1$, $mach_2$, $\cdots$, $mach_n$, respectively. We define `next_event` for the subclass synchronous product as follows.

```
next_event ($mach$, $Ev$) :
        returns   event label $Ev = \sigma$ if $\sigma \in \bigcup_{i=1}^{n} \Sigma_i$, and
                  $\forall\ i,\ 1 \le i \le n,\ \sigma \in \Sigma_i$, next_event ($mach_i$, $\sigma$) is true;
          fails   otherwise.
```

Now, we can define `ntran` for synchronous product as follows.

```
ntran ($mach$, $Ev$, $En$) :                                                        (3.4.1)
        returns   $Ev = \sigma$ and $En = [state_1,\ state_2,\ \cdots,\ state_n]$ if
                  next_event ($mach$, $\sigma$) is true, and
                  ($\forall\ i,\ 1 \le i \le n$, either
                  (ntran ($mach_i$, $\sigma$, $state_i$) is true), or
                  (next_event($mach_i$, $\sigma$) is false and state($mach_i$, $state_i$) is true));
          fails   otherwise.
```

3.5 Shuffle Products

Finally, for the shuffle product, no synchronization of events is necessary. Hence, we have

```
ntran ($mach$, $Ev$, $En$) :                                                        (3.5.1)
        returns   $Ev = \sigma$and $En = [state_1,\ state_2,\ \cdots,\ state_n]$ if
```

$\exists\, i,\ 1 \leq i \leq n,$ ntran $(mach_i,\ \sigma,\ state_i)$ is true and

$\forall\, j,\ 1 \leq j \leq n,\ j \neq i,$ `state` $(mach_j,\ state_j)$ is true;

fails otherwise.

4.0 Implementation

For building a large-scale discrete-event system simulation, we have defined subclasses of DES to model various flat and architectural representations of finite state machines. We proceed now to investigate the inter-object relationships and the issues related to parallel and distributed processing.

4.1 Object-Oriented Implementation

An object-oriented implementation of a product structure is shown in Fig 2. Object *mach* is an instance of Class CP and has two children $mach_1$ and $mach_2$ from the Class FSM. Object *mach* does not have a local database to process the incoming command **ntran** and it must *delegate* the command to its children via message-passing according to the method defined in (3.3.1).

The processing of the command **ntran** is different: *mach* processes it via delegation, and $mach_1$ and $mach_2$ via invoking their encapsulated local databases. This adaptive processing of commands according to the recipient of the command is known within the object-oriented paradigm as *polymorphism*. Polymorphism is a useful software engineering feature for handling large-scale systems since the designer does not need to know the class information of the recipient objects.

The transition structure of the Cartesian product is computed 'just-in-time' in the sense that *mach* does not compute and store its transition structure at its instantiation time. The advantage of this just-in-time approach is that the storage requirement of a product structure grows linearly according to the number of children, where a flat representation (by computing the FSM representation of the Cartesian product) grows exponentially. This advantage is gained at the price of greater computational effort because *mach* must compute **ntran** $(mach,\ Ev,\ En)$

afresh every time the command is invoked because it does not have an internal data structure to store the previous results.

4.2 Parallel and Distributed Processing

A command can be processed locally by an architectural object such as one instantiated from a product class if all its children use local databases. Consider the architecture used for computing a supremal controllable sublanguage in the Ramadge-Wonham framework [5] as shown in Fig. 3. The processing of *mach_spec* can be partitioned using three processors by grouping *mach*, $mach_1$ and $mach_1$ into one, and *spec*, $spec_1$ and $spec_2$ into the other, and *mach_spec* into the third. Object *mach_spec* is then interfaced with *mach* and *spec* via Branches 1 and 2.

From the definitions of the kernel set of commands for the product structures, the order of execution is immaterial. Hence the child commands can be executed in parallel as long as there are no interconnecting branches between the child objects. Take, for example, again the architecture in Fig. 3. If a `ntran` command is sent to the object *mach_spec*, the command can be delegated to *mach* and *spec* in parallel. Objects *mach* and *spec* can in turn delegate the same command in parallel to their children. The speed-up gained by parallel processing of command should make our 'just-in-time' strategy attractive if not essential for handing large-scale discrete-event systems.

REFERENCES

[1] W.M. Wonham, "A control theory for discrete-event systems," *NATO ASI Series*, Vol. F47, Springer-Verlag, 1988.

[2] C.H. Golaszewski and P.J. Ramadge, "Control of discrete event processes with forced events," *Proc. 26nd Conf. Decision Contr.*, pp. 247-251, Dec. 1987.

[3] H. Zhong and W.M. Wonham, "On hierarchical control of discrete-event system systems," *Proc. 1988 Conference on Information Sciences and Systems, Dept. of Electrical Engineering, Princeton University*, pp. 64-70, 1988.

[4] C.A.R. Hoare, *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[5] W.M. Wonham and P.J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. and Optimization*, Vol 25(3), pp. 637-659, 1987.

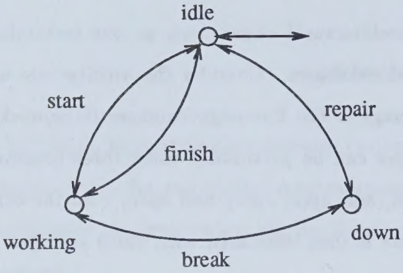

idle
start
repair
finish
working
down
break

Fig. 1 A Simple Machine

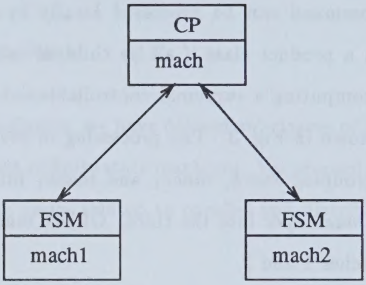

CP
mach
FSM
mach1
FSM
mach2

Fig.2 A Product Structure

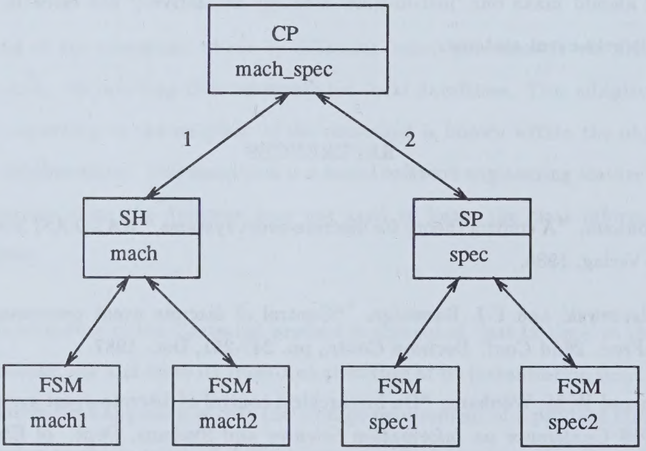

CP
mach_spec
1
2
SH
mach
SP
spec
FSM
mach1
FSM
mach2
FSM
spec1
FSM
spec2

Fig. 3 A Supervisor Synthesis Structure

Stability of Sampled-Data Feedback Systems[1]

TONGWEN CHEN and BRUCE A. FRANCIS
Department of Electrical Engineering
University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract. This paper considers the stability of MIMO sampled-data feedback systems. Suppose a continuous-time LTI feedback system is internally stable. It is shown that the hybrid system with the digital implementation of the controller is exponentially stable if the sampling period is small enough and is $\mathcal{L}_p$ input-output stable ($1 \leq p \leq \infty$) if furthermore each sampler is associated with a suitable low-pass pre-filter. The next result relates exponential stability to input-output stability of a sampled-data system composed of a continuous LTI plant and a discrete time-varying controller with a sampler and a zero-order hold. For the system to be $\mathcal{L}_p$ input-output stable ($1 \leq p < \infty$), it's necessary to include a low-pass pre-filter in the loop. It can then be proved that $\mathcal{L}_p$ stability of the closed-loop sampled-data system follows from its exponential stability.

1 Introduction

A sampled-data feedback system is a continuous-time system (the plant) in feedback connection with a discrete-time system (the controller) through a sampler and a hold device. Suppose that the sampler is periodic and the hold device is synchronous in time with the sampler. Then even if the plant and the controller of the sampled-data system are linear time-invariant (LTI), the overall system is in general not time-invariant due to the time-varying characteristics of the sampler. Conventionally, one is concerned with the behavior of a sampled-data system only at its sampling instants. In this case, the system becomes LTI, and hence various discrete-time control techniques could be applied in the design (e.g. [1] and [2]). Along this direction, more intricate control and sampling schemes have been developed, e.g. the discrete-time pole assignment results by periodically time-varying output feedback [3, 4], and by a multi-rate sampling mechanism [5, 6]. In contrast, the treatment of sampled-data systems by Morari and Zafiriou in [7] is different. There the intersample behavior is considered and performance of sampled-data systems is defined in terms of the continuous plant outputs.

Discrete-time stability of the closed-loop system implies, under certain conditions, exponential stability of the autonomous hybrid system [8]; however, in general it does not imply $\mathcal{L}_2$ input-output stability. The difficulty here is that the sampler does not map every continuous function in $\mathcal{L}_2$ into $\mathcal{S}_2$, the space of sequences which are square summable. Then the question arises, when does exponential stability of the hybrid system imply $\mathcal{L}_2$ input-output stability?

One motivation for this paper is digital implementation of $\mathcal{H}_\infty$ optimal controllers. The $\mathcal{H}_\infty$ norm is the induced norm for inputs and outputs in $\mathcal{L}_2$, so it would be desirable to know when such a digital implementation yields an $\mathcal{L}_2$ stable closed-loop system.

Thompson *et al.* [9] attacked the $\mathcal{L}_2$ stability problem of sampled-data feedback systems using conic sector analysis, which was first introduced by Zames [10]. Their result involves the construction of a center operator and a radius operator for the cone in the analysis. Though the center can be chosen to be the LTI approximation to the sampled-data operator which they called "the optimal center", how to determine the radius so that closed-loop stability is guaranteed is still in question. Another problem of their analysis is that it can be applied

---

[1] This work has been supported by the Natural Sciences and Engineering Research Council of Canada.

only to systems with *stable* digital controllers. Some effort has been made to remove this restriction [11], but the choice of radius operator in general remains unsolved.

Unlike the conic sector approach, which is basically in the frequency domain, our analysis will be conducted in the time domain. The setup is similar to that found in [8]. For a detailed derivation and the proofs of results, see the full version of this paper [12].

Finally, we introduce some notation. Script capital letters represent function spaces. The norm symbol $\|\cdot\|$ is the Euclidean norm if applied to a vector in $\mathbb{C}^n$, the largest singular value if applied to a matrix in $\mathbb{C}^{n\times m}$, the induced norm if applied to an operator. The spectrum of a matrix A in $\mathbb{C}^{n\times n}$ is denoted $\sigma(A)$. The open left-half complex plane is $\{\text{Re } s < 0\}$. The open unit disk is $\{|\lambda| < 1\}$. Transfer matrices are represented in terms of their state-space realizations. Thus, in continuous time,

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] := D + C(sI - A)^{-1}B$$

in discrete time,

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] := D + \lambda C(I - \lambda A)^{-1}B$$

Here we use λ instead of $z = \lambda^{-1}$.

2 Sample and hold transformations

The purpose of this section is to establish some preliminary results on the sample and hold operators for the development to follow.

Let $\mathcal{C}(\mathbb{C}^n)$ be the vector space of piecewise-continuous functions from the time set $[0, \infty)$ to $\mathbb{C}^n$ that are bounded on compact subsets of $[0, \infty)$ and are continuous from the left at every point except the origin. Usually the dimension n is irrelevant, so we simply write $\mathcal{C}$. Let $\mathcal{L}_p(\mathbb{C}^n)$ $(1 \le p \le \infty)$ denote the *Lebesgue* space of measurable functions $f(t)$ from $[0, \infty)$ to $\mathbb{C}^n$ which have finite p-norm, i.e.

$$\|f(t)\|_p := \left(\int_0^\infty \|f(t)\|^p \, dt\right)^{\frac{1}{p}} < \infty$$

Then the space $\mathcal{C}_p(\mathbb{C}^n)$ is the intersection of $\mathcal{L}_p(\mathbb{C}^n)$ and $\mathcal{C}(\mathbb{C}^n)$. As a subspace of $\mathcal{L}_p(\mathbb{C}^n)$, $\mathcal{C}_p(\mathbb{C}^n)$ is equipped with the $\mathcal{L}_p$-norm. Again, the spaces $\mathcal{L}_p(\mathbb{C}^n)$ and $\mathcal{C}_p(\mathbb{C}^n)$ are abbreviated $\mathcal{L}_p$ and $\mathcal{C}_p$.

Similarly in discrete time, let $\mathcal{S}(\mathbb{C}^n)$ denote the vector space of $\mathbb{C}^n$-valued sequences on the time set $\{0, 1, 2, \cdots\}$. The subspace $\mathcal{S}_p(\mathbb{C}^n)$ of $\mathcal{S}(\mathbb{C}^n)$ consists of all sequences ξ such that

$$\left(\sum_{k=0}^{\infty} \|\xi(k)\|^p\right)^{\frac{1}{p}} < \infty$$

The left-hand side, also denoted $\|\xi\|_p$, is defined to be the norm in $\mathcal{S}_p(\mathbb{C}^n)$. Also, these are abbreviated $\mathcal{S}$ and $\mathcal{S}_p$ if the context is clear.

Now we define two linear transformations. The sampler with period h,

$$\begin{aligned} S_h: \ \mathcal{C} \longrightarrow \mathcal{S}, \qquad & y = S_h u \\ y(0) = 0 & \\ y(k) = u(kh) \qquad & (k \ge 1) \end{aligned}$$

and the zero-order hold,

$$\begin{aligned} H_h: \ \mathcal{S} \longrightarrow \mathcal{C}, \qquad & w = H_h v \\ w(0) = v(0) & \\ w(t) = v(k) \qquad & (kh < t \le (k+1)h) \end{aligned}$$

If we restrict the domains of the two transformations, we get that $S_h : \mathcal{C}_\infty \longrightarrow \mathcal{S}_\infty$ is a bounded operator with induced norm 1 and so is $H_h : \mathcal{S}_p \longrightarrow \mathcal{C}_p$ with norm $h^{1/p}$ $(1 \leq p \leq \infty)$. Note also that $S_h H_h$ is a bounded operator on $\mathcal{S}_p$ $(1 \leq p \leq \infty)$ with norm 1; in fact, it's the backward shift operator (unit delay).

The situation with the sample and hold transformation $H_h S_h$ is more delicate and interesting. Since $S_h : \mathcal{C}_\infty \longrightarrow \mathcal{S}_\infty$ and $H_h : \mathcal{S}_\infty \longrightarrow \mathcal{C}_\infty$ are both bounded, it follows that $H_h S_h$ is bounded on $\mathcal{C}_\infty$; actually the norm $\|H_h S_h\|$ is 1 on $\mathcal{C}_\infty$. However, the sample and hold transformation is not a bounded operator on $\mathcal{C}_p$ $(1 \leq p < \infty)$. Consider the function $u(t)$ defined by

$$u(t) = \sum_{k=1}^{\infty} v_k(t) \tag{1}$$

where

$$v_k(t) = \begin{cases} 1 - \dfrac{2k^2}{h}|t - kh| & \text{if } |t - kh| < \dfrac{h}{2k^2} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

It's straightforward to verify that $u(t)$ is in $\mathcal{C}_p$ but $y := H_h S_h u$ is not in $\mathcal{C}_p$ for any $1 \leq p < \infty$ since $y \equiv 1$ for $t > h$. Thus $\mathcal{C}_p$ is not an admissible domain for the sample and hold transformation.

It has been shown that the sample and hold transformation is bounded on the class of bandlimited $\mathcal{L}_2$ signals if the sampling frequency is higher than the Nyquist frequency [13]. So $H_h S_h$ with a suitable *ideal* low-pass pre-filter is a bounded operator on $\mathcal{L}_2$. For practical reasons and also for the results to follow, we'll show that the ideal low-pass filter can be replaced by any finite-dimensional, strictly causal, stable LTI system and that the boundedness result can be extended to any space $\mathcal{L}_p$ $(1 \leq p < \infty)$ for any sampling frequency.

Let E be a set. The characteristic function of the set E is denoted χ_E. Let $\mathcal{L}_p^e$ denote the extended $\mathcal{L}_p$ space, i.e., the space of measurable functions $f(t)$ on $[0, \infty)$ such that $\chi_E f \in \mathcal{L}_p$ for any subset E of $[0, \infty)$ with finite measure. It follows from Hölder's inequality that $\mathcal{L}_p^e$ is contained in $\mathcal{L}_1^e$ for every $1 < p \leq \infty$; hence so is $\mathcal{L}_p$.

The following result from the interpolation of operators is needed.

Lemma 1 *Suppose T is a linear transformation on $\mathcal{L}_1^e$. If T maps $\mathcal{L}_1$ to $\mathcal{L}_1$ and $\mathcal{L}_\infty$ to $\mathcal{L}_\infty$ and the restrictions are bounded operators with norms M_1 and M_∞ respectively, then T maps $\mathcal{L}_p$ to $\mathcal{L}_p$ and the restriction is a bounded operator with norm no greater than $M_1^{1/p} M_\infty^{1/q}$ where $\frac{1}{p} + \frac{1}{q} = 1$.*

This is a special case of the M. Riesz convexity theorem (e.g. [15]).

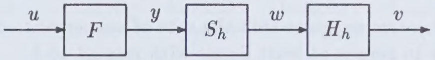


Figure 1

Now consider the configuration in Figure 1, the sample and hold transformation with a pre-filter F. Assume F is a finite-dimensional, strictly causal, stable LTI system with a stabilizable and detectable state-space description

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t)$$

The pre-filter F maps $\mathcal{L}_1^e$ to $\mathcal{C}$; specifically, it takes any $\mathcal{L}_p$ space into the space $\mathcal{C}_p$. By a result in [9], we know that the transformation $H_h S_h F$ is bounded on $\mathcal{L}_2$.

Theorem 1 *Pertaining to Figure 1, the sample and hold transformation with a pre-filter* H_hS_hF *is a bounded operator on* $\mathcal{L}_p$ *for all* $1 \le p \le \infty$.

The result on boundedness of H_hS_h on bandlimited $\mathcal{L}_2$ signals in [13] can be deduced from our Theorem 1.

Lemma 2 *Let* T *be a linear transformation from* $\mathcal{C}_2$ *to* $\mathcal{L}_2^e$. *Let* F_{id} *be an ideal band-pass filter from* $\mathcal{L}_2$ *to* $\mathcal{C}_2$ *with passband* $\Omega := [\omega_1, \omega_2]$ *and* F *a square pre-filter as before such that it has no imaginary zeros on the passband of* F_{id}. *If the mapping* TF *is bounded on* $\mathcal{L}_2$, *then the mapping* TF_{id} *is also bounded on* $\mathcal{L}_2$ *and*

$$\|TF_{id}\| \le \max_{\omega \in \Omega} \|\hat{F}^{-1}(j\omega)\| \cdot \|TF\|$$

where $\hat{F}$ *denotes the Fourier transform of the impulse response of* F.

Proof It's a fact that F_{id} is a bounded operator on $\mathcal{L}_2$ with induced norm 1. Thus by the hypothesis the mapping TFF_{id} is bounded on $\mathcal{L}_2$ and hence on the subspace $\mathcal{B} := F_{id}\mathcal{L}_2$, namely, the space of $\mathcal{L}_2$ functions with band limited to Ω. The operator FF_{id} maps $\mathcal{B}$ to $\mathcal{B}$ and the restriction $FF_{id}|\mathcal{B}$ has a bounded inverse since $\hat{F}(j\omega)$ is invertible over the frequency band Ω. Let $T|_{\mathcal{B}}$ denote the mapping T when the domain space is restricted to $\mathcal{B}$. Then

$$T|_{\mathcal{B}} = (TFF_{id})(FF_{id}|\mathcal{B})^{-1}$$

So $TF_{id} = T|_{\mathcal{B}}F_{id}$ is bounded on $\mathcal{L}_2$ and

$$\|TF_{id}\| \le \|T|_{\mathcal{B}}\| \le \|(FF_{id}|\mathcal{B})^{-1}\| \cdot \|TFF_{id}\| \le \max_{\omega \in \Omega} \|\hat{F}^{-1}(j\omega)\| \cdot \|TF\| \quad \blacksquare$$

It follows from Theorem 1 and Lemma 2 that the sample and hold transformation with any ideal band-pass filter $H_hS_hF_{id}$ is bounded on $\mathcal{L}_2$. Note that we do not assume that the sampling frequency is higher than the Nyquist frequency. Since the impulse response of any ideal band-pass filter F_{id} does not belong to $\mathcal{L}_1$, the mapping F_{id} is neither bounded on $\mathcal{L}_\infty$ nor on $\mathcal{L}_1$. Hence $H_hS_hF_{id}$ is unbounded on either $\mathcal{L}_\infty$ or $\mathcal{L}_1$.

Theorem 1 implies that the sampler with a pre-filter is well-behaved in the sense that it is a bounded operator from $\mathcal{L}_p$ to $\mathcal{S}_p$ for all $1 \le p \le \infty$. Since F is also a bounded operator on $\mathcal{L}_p$, it will be of interest to ask whether the operator H_hS_hF approaches F in some way when the sampling period h goes to zero.

Theorem 2 *The operator* H_hS_hF *in Figure 1 converges to* F *as* h *approaches zero in the* $\mathcal{L}_p$ *induced norm for all* $1 \le p \le \infty$.

The proof of this theorem suggests that the rate of convergence of $\|(I - H_hS_h)F\|$ as the sampling period h goes to zero is at least linear with respect to h.

From Theorem 2 and Lemma 2 we can also deduce that $H_hS_hF_{id}$ converges to F_{id} as h goes to zero in the $\mathcal{L}_2$ induced norm. The rate of convergence is also at least linear with respect to h.

3 Digital implementation of continuous-time controllers

Consider a basic feedback control system with a continuous-time plant and controller, designed so that closed-loop stability is achieved. In many cases it's desirable to implement the controller digitally. A natural way to get a digital controller is by discretizing the continuous-time counterpart. The purpose of this section is to derive results on stability of such sampled-data feedback systems.

Consider an autonomous, continuous-time, finite-dimensional system. A state at time t, $x(t)$, of the system is sufficient information for the computation of all future values of all signals. By *exponential stability* we mean that there exist positive constants α and β such that for every initial time t_0 and initial state $x(t_0)$

$$\|x(t)\| \le \|x(t_0)\|\beta e^{-\alpha(t-t_0)} \qquad (t \ge t_0)$$

Similar definition can be stated in discrete time.

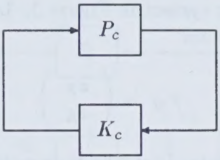


Figure 2

In the continuous-time feedback configuration of Figure 2, assume both the plant P_c and the controller K_c are finite-dimensional, LTI and strictly causal. Bring in stabilizable and detectable state-space realizations

$$P_c = \left[\begin{array}{c|c} A_P & B_P \\ \hline C_P & 0 \end{array}\right] \qquad K_c = \left[\begin{array}{c|c} A_K & B_K \\ \hline C_K & 0 \end{array}\right]$$

If we choose the compound of the states for P_c and K_c to be a state for the closed-loop system, then the A-matrix of the closed-loop state-space model is

$$\bar{A}_c = \begin{bmatrix} A_P & B_P C_K \\ B_K C_P & A_K \end{bmatrix}$$

It follows that the closed-loop system $(P_c,\ K_c)$ is exponentially stable iff the matrix $\bar{A}_c$ is stable, namely, $\sigma(\bar{A}_c) \subset \{\text{Re } s < 0\}$. Similar conclusion holds in the discrete-time setting.

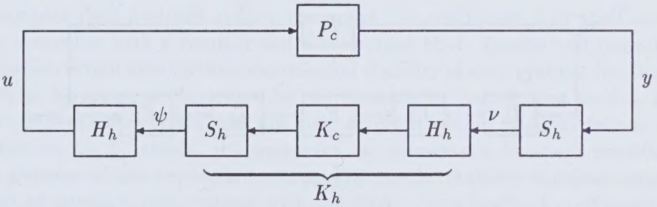


Figure 3

Now consider a digital implementation of the controller K_c: include sample and zero-order hold devices into the feedback loop as shown in Figure 3, where $K_h := S_h K_c H_h$ can be regarded as a digital controller. If we consider the behavior of the system only at its sampling instants, Figure 3 is equivalent to a standard discrete-time feedback loop with the plant $P_h := S_h P_c H_h$ and the controller K_h (functions of the sampling period h):

$$P_h = \left[\begin{array}{c|c} e^{A_P h} & \int_0^h e^{A_P \tau}\, d\tau B_P \\ \hline C_P & 0 \end{array}\right] \qquad K_h = \left[\begin{array}{c|c} e^{A_K h} & \int_0^h e^{A_K \tau}\, d\tau B_K \\ \hline C_K & 0 \end{array}\right]$$

Theorem 3 *If $(P_c,\, K_c)$ is exponentially stable, there exists a positive number δ such that $(P_h,\, K_h)$ is also exponentially stable for every sampling period h in the interval $(0,\, \delta)$.*

Notice that in our framework we assume P_c and K_c are both strictly causal. This assumption is necessary for Theorem 3 to hold. In Figure 2 if we set $P_c = k_1$ and $K_c = k_2$, where k_1 and k_2 are constants satisfying $|k_1 k_2| > 1$, then $(P_c,\, K_c)$ is stable but $(P_h,\, K_h)$ is not stable for any h. This is due to the fact that $S_h H_h$ is the backward shift operator, a consequence of our definitions for the sampler and zero-order hold.

Now consider the *sampled-data* system in Figure 3. Let x_P and x_K be the state vectors for P_c and K_c respectively. The vector

$$x_{sd} := \begin{pmatrix} x_P \\ x_K \end{pmatrix}$$

is a state for the sampled-data system. It can be shown that for any $t > 0$, $kh < t \leq (k+1)h$, knowledge $x_{sd}(t)$ allows computation of all continuous signals at time t and all discrete signals at the $(k+1)$th sampling instant and of the vector $x_{sd}(t_1)$ for all $t_1 > t$; this can be seen by writing out all equations associated with the hybrid system.

The following result on exponential stability of the hybrid system is obtained.

Theorem 4 *If $(P_c,\, K_c)$ is exponentially stable, there exists a positive number δ such that the hybrid system in Figure 3 is also exponentially stable for every $h \in (0,\, \delta)$.*

Proof This is a consequence of Theorem 3 above and Theorem 4 in [8]. ■

Now let's turn to the input-output stability analysis of our hybrid system shown in Figure 4, where $T_h := H_h S_h$. Assume P_c and K_c are *linear transformations* on $\mathcal{L}_1^e$ and the associated pre-filters F_1, F_2 are finite-dimensional, LTI, stable and strictly causal. By $\mathcal{L}_p$ *stability* of this configuration we mean that the linear transformation from $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ to $\begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ is a bounded operator on $\mathcal{L}_p$.

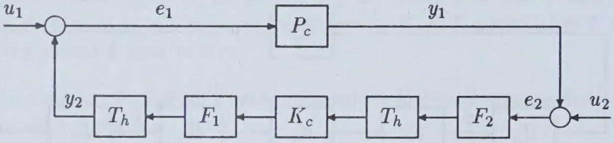


Figure 4

For the hybrid system to be $\mathcal{L}_p$ stable $(1 \leq p < \infty)$, the presence of F_2 is necessary. This can be illustrated by an example. Suppose F_2, P_c and K_c in Figure 4 are identity transformations. Let the transfer function for F_1 be $-\frac{1}{s+1}$. It can be verified that the hybrid system is exponentially stable for all sampling periods $h > 0$. However, for any fixed h, apply the inputs $u_1 = 0$, $u_2 = u \in \mathrm{C}_p$, where u is given in equations (1) and (2). On observing that the sampled u is a unit step signal, we can easily compute the steady-state value $y_1 = -\frac{1}{2}$. Thus the hybrid system is *not* $\mathcal{L}_p$ stable. The inclusion of F_1 in the feedback loop is merely for technical reasons.

The original continuous-time system corresponding to the hybrid system in Figure 4 is pictured in Figure 5, in which we have introduced another exogenous input u_3. Such a configuration is said to be $\mathcal{L}_p$ *stable* if the mapping

$$\Psi : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \longrightarrow \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix}$$

is a bounded operator on $\mathcal{L}_p$.

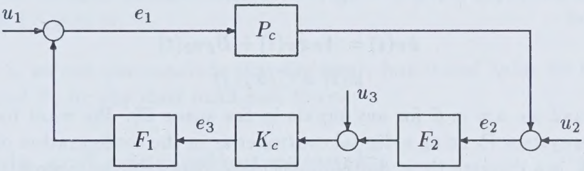


Figure 5

Theorem 5 *Assume P_c and K_c are general linear transformations on $\mathcal{L}_1^e$. For any $1 \le p \le \infty$, if the continuous-time system in Figure 5 is $\mathcal{L}_p$ stable, there exists a positive number δ such that the corresponding hybrid system in Figure 4 is also $\mathcal{L}_p$ stable for every $h \in (0,\ \delta)$.*

If furthermore our plant P_c and controller K_c are finite-dimensional and LTI, so is the system in Figure 5. It follows that $\mathcal{L}_p$ stability ($1 \le p \le \infty$) of the system is implied by its exponential stability [16]. The following result can be obtained from Lemma 1 and Theorem 5.

Corollary 1 *Assume P_c and K_c are finite-dimensional, LTI and causal. If the continuous-time system in Figure 5 is exponentially stable, there exists a positive number δ such that for every $h \in (0,\ \delta)$ the hybrid system in Figure 4 is $\mathcal{L}_p$ stable for all $1 \le p \le \infty$.*

4 Sampled-data feedback systems

Consider a sampled-data feedback system composed of a continuous LTI plant and a discrete time-varying controller with a sampler and a zero-order hold. Francis and Georgiou [8] have shown that under certain assumptions exponential stability of such systems implies $\mathcal{L}_\infty$ input-output stability. However, as illustrated by our example in the previous section, exponential stability of a sampled-data system does not yield $\mathcal{L}_p$ input-output stability for any $1 \le p < \infty$. For the system to be $\mathcal{L}_p$ stable, it's necessary to introduce a low-pass pre-filter into the system. The purpose of this section is to relate exponential stability to input-output stability in the context of sampled-data systems with pre-filters. Two feedback configurations will be examined.

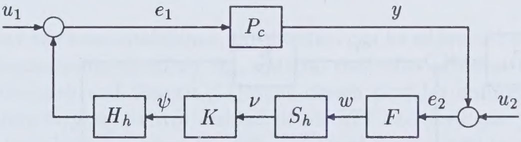


Figure 6

Figure 6 shows a sampled-data control system with a filter F preceding the sampler. This is the common configuration in practice, F being an anti-aliasing filter.

Pertaining to the hybrid system shown in Figure 6, assume the plant P_c is a continuous-time, finite-dimensional, LTI system which has a state model

$$\dot{x}_P(t) = A_P x(t) + B_P e_1(t)$$

$$y(t) = C_P x_P(t) + D_P e_1(t)$$

As usual, the pre-filter is assumed to be finite-dimensional, LTI, stable and strictly causal with state model

$$\dot{x}_F(t) = A_F x_F(t) + B_F e_2(t)$$

$$w(t) = C_F x_F(t)$$

The states x_P and x_F are in $\mathcal{C}$ for any inputs in the space $\mathcal{L}_1^e$. We want to control a given continuous-time system P_c using a digital controller as in the configuration of Figure 6. Here the controller K is a discrete-time, finite-dimensional, time-varying system with state model

$$\xi(k+1) = A_k \xi(k) + B_k \nu(k)$$

$$\psi(k) = C_k \xi(k) + D_k \nu(k)$$

where A_k, B_k, C_k, D_k are uniformly bounded in k. As defined in Section 2, S_h is the sampler with sampling period h and H_h the zero-order hold.

Note that if A_k, B_k, C_k, D_k are periodic in k, they are uniformly bounded with respect to k. Thus our class of digital controllers includes all the finite-dimensional periodic ones.

To consider exponential stability of the hybrid system in Figure 6, let u_1, u_2 be zero. Recall that a state at time t is sufficient information to compute all future values of all signals, namely, the continuous signals x_P, x_F, e_1, y, w and the discrete signals ξ, ν, ψ. It can be shown [12] that a state for the sampled-data system is

$$x_{sd} := \begin{pmatrix} x_P \\ x_F \\ H_h \psi \\ H_h U^* \xi \end{pmatrix}$$

where U^* is the forward shift operator on $\mathcal{S}$.

Theorem 6 *Pertaining to the configuration of Figure 6, if the hybrid system is exponentially stable, so is it $\mathcal{L}_p$ stable for all $1 \le p \le \infty$.*

Another configuration of interest is given in Figure 7. Here the plant P_c, controller K, pre-filters F_1 and F_1 are subject to the same assumptions made before. The new feature of this configuration is that the low-pass pre-filters are placed outside the feedback loop. We view these filters as being fictitious; their function is to generate a class of input signals with attenuated high frequency component. As we shall see, an anti-aliasing filter is not required for stability in this case.

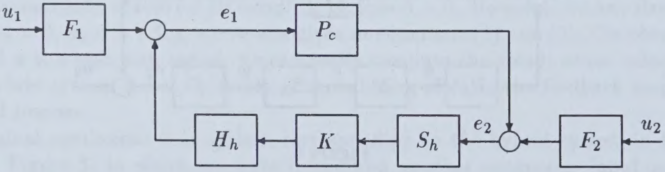


Figure 7

Such a feedback configuration is said to be $\mathcal{L}_p$ *stable* if the mapping from $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ to $\begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$ is bounded on $\mathcal{L}_p$. Since the pre-filters F_1 and F_2 are stable, it follows from [8] that exponential stability implies $\mathcal{L}_\infty$ stability. The $\mathcal{L}_1$ stability of such system can be established from its exponential stability. Thus Lemma 1 allows us to state the following result.

Theorem 7 *Pertaining to the hybrid system in Figure 7, if it is exponentially stable, so is it* $\mathcal{L}_p$ *stable for all* $1 \le p \le \infty$.

By Lemma 2, we can also conclude that the result just stated holds for the case $p = 2$ if we replace F_1 and F_2 by any ideal band-pass filters.

5 Example and concluding remarks

To illustrate our result in Section 3, consider a standard control system configuration with unit feedback. The plant P is an experimental flexible beam at the University of Toronto with transfer function

$$P(s) = \frac{1.6188s^2 - 0.1575s - 43.9425}{s^4 + 0.1736s^3 + 27.9001s^2 + 0.0186s}$$

The stabilizing controller K is designed using an $\mathcal{H}_\infty$ optimization method:

$$K(s) = \frac{-0.0460s^5 - 1.5402s^4 - 1.5498s^3 - 42.75s^2 - 0.0285s - 1.58 \times 10^{-4}}{s^6 + 3.766s^5 + 34.9509s^4 + 106.2s^3 + 179.2s^2 + 166.43s + 0.0033}$$

From the Bode plot of PK, $|PK(j\omega)|$ equals 1 at $\omega = 0.4$ and is less than 1 beyond this frequency. Insert a pre-filter $F_2(s) = \frac{1}{s+1}$ preceding K. By the Nyquist stability criterion, the closed-loop system remains stable. Redraw the system diagram as in Figure 5 with $P_c = -P$, $F_1 = F_2 = \frac{1}{s+1}$, $K_c(s) = (s+1)K(s)$. Note that $F_1 K_c = K$. We view F_1 as being fictitious; its role is actually performed by the strictly causal controller K.

Suppose we want the sampled version of the control system as shown in Figure 4 to be $\mathcal{L}_2$ stable. The intermediate results in the proofs of Theorems 2 and 5 can be used to estimate the upper bound for our sampling period:

$$\delta \approx 10^{-4} \text{ sec.}$$

For details of this computation see [12].

To maintain $\mathcal{L}_2$ stability in the digital implementation, our estimated upper bound for the sampling period may tend to be conservative. However, it provides information which guarantees stability of the sampled version of our flexible beam control system. To get a less conservative result, we need to find a better bound, or if possible the norm, for the operator $(H_h S_h - I)F$.

Once $\mathcal{L}_2$ stability has been established, the problem can be addressed of how fast to sample so that a digital implementation of an $\mathcal{H}_\infty$ optimal controller achieves a pre-specified level of performance. Feintuch and Francis [17] have shown that for uniformly optimal control problems, using time-varying control instead of an LTI one does not yield any improved performance. So digital implementation of $\mathcal{H}_\infty$ optimal control will in general degrade the performance. How to quantify this degradation as a function of the sampling period is an important subject for further research.

References

[1] J. R. Ragazzini and G. F. Franklin, *Sampled-Data Control Systems*, McGraw-Hill Book Company, Inc., New York, 1958.

[2] K. J. Aström and B. Wittenmark, *Computer Controlled Systems: Theory and Design*, Prentice-hall, Englewood Cliffs, N. J., 1984.

[3] A. B. Chammas and C. T. Leondes, On the design of linear time invariant systems by periodic output feedback, part 1: discrete-time pole assignment, Int. J. Control, vol. 27, No. 6, pp 885-894, 1978.

[4] A. B. Chammas and C. T. Leondes, Pole assignment by piecewise constant output feedback, Int. J. Control, vol. 29, No. 1, pp 31-38, 1979.

[5] M. Araki and T. Hagiwara, Pole assignment by multirate sampled-data output feedback, Int. J. Control, vol. 44, pp 1661-1673, 1986

[6] T. Hagiwara and M. Araki, Design of a stable feedback controller based on the multirate sampling of the plant output, IEEE Trans. Automat. Control, vol. 33, No. 9, pp 812-819, 1988.

[7] M. Morari and E. Zafiriou, *Robust Process Control*, Prentice-Hall, Englewood, Cliffs, N. J., 1989.

[8] B. A. Francis and T. T. Georgiou, Stability theory for linear time-invariant plants with periodic digital controllers, IEEE Trans. Automat. Control, vol. 33, No. 9, pp 820-832, 1988.

[9] P. M. Thompson, G. Stein and M. Athans, Conic sectors for sampled-data feedback systems, Systems & Control Letters, vol. 3, pp 77-82, 1983.

[10] G. Zames, On the input-output stability of time varying nonlinear feedback systems (part 1), IEEE Trans. Automat. Control, vol. 11, No. 2, pp 228-238, 1966.

[11] P. M. Thompson, R. L. Dailey and J. C. Doyle, New conic sectors for sampled-data feedback systems, Systems & Control Letters, vol. 7, pp 395-404, 1986.

[12] T. Chen and B. A. Francis, Stability of sampled-data systems, Systems Control Group Report No. 8905, Dept. of Elec. Eng., Univ. of Toronto, Toronto, Canada, May, 1989.

[13] T. Perry, On the recovery of $\mathcal{H}_\infty$ bounds in sampled-data systems, M.A.Sc. thesis, Dept. of Elec. Eng., Univ. of Toronto, Toronto, Canada, 1989.

[14] H. L. Royden, *Real Analysis*, MacMillan, New York, 1968.

[15] E. M. Stein and G. Weiss, *Introduction to Fourier Analysis on Euclidean Spaces*, Princeton University Press, Princeton, N. J., 1971.

[16] M. Vidyasagar, *Nonlinear System Analysis*, Prentice-Hall, Englewood Cliffs, N. J., 1978.

[17] A. Feintuch and B. A. Francis, Uniformly optimal control of linear feedback systems, Automatica, vol. 21, No. 5, pp 563-574, 1985.

ASYMPTOTIC STATISTICAL PROPERTIES OF LINEAR SYSTEMS OPERATING UNDER QUANTIZED FEEDBACK

David F. Delchamps
School of Electrical Engineering
Cornell University
Phillips Hall
Ithaca, NY 14853

ABSTRACT

This paper considers the asymptotic behavior of a time-invariant discrete-time linear system operating under the influence of feedback control laws that are nonanticipative functions of quantized measurements of the system's state. It is seen that treating quantized measurements as noisy observations can lead one to draw incorrect conclusions about the behavior of closed loop systems obtained from "naive" implementations of standard linear feedback gains. Modeling quantization more realistically as a memoryless nonlinearity enables one to apply some powerful machinery from the ergodic theory of dynamical systems to the careful analysis of the feedback systems under consideration. One finds that their closed loop dynamics can be exceedingly complicated; even when the open loop and closed loop systems are both nominally stable, there can exist an arbitrarily large number of stable closed loop equilibria or periodic orbits. Moreover, if the open loop system is unstable and the closed loop system is stable, there exists an invariant measure supported on a bounded subset of the state space with respect to which the closed loop dynamics have a finite ergodic decomposition; there are also in this case finitely many Cantor sets invariant under the closed loop dynamics. The size of the set upon which the measure is concentrated depends on the parameters of the open and closed loop systems; under certain conditions, the measure is absolutely continuous with respect to Lebesgue measure and differs substantially from the invariant measure that a signal-plus-noise analysis would predict. It is quite difficult in general to compute the invariant measure exactly, but excellent approximations can be obtained by iterating the Perron-Frobenius operator corresponding to the closed-loop system.

1. INTRODUCTION

Consider the time-invariant discrete-time state space linear system governed by the difference equation

$$x(k+1) = Ax(k)+Bu(k)\ , \quad k \geq 0\ , \qquad (*)$$

where the real matrices A and B have respective sizes $(n \times n)$ and $(n \times m)$. Standard linear system theory tells us that when (A,B) is a reachable pair, there exists for any choice of n complex numbers (closed under conjugation) a real $(m \times n)$ matrix F that makes those numbers the eigenvalues of $A-BF$. In particular, it is always possible to find an F for which all the eigenvalues of $A-BF$ lie in the unit disc $|z| < 1$, in which case $\bar{x} = 0$ is a globally asymptotically stable equilibrium for the "closed loop" difference equation

$$x(k+1) = (A-BF)x(k)\ , \quad k \geq 0\ ,$$

that one obtains after setting $u(k) = -Fx(k)$, $k \geq 0$.

The implementability of such linear state feedback control laws is contingent upon one's ability to measure $x(k)$, $k \geq 0$, exactly. Such a fortuitous state of affairs exists only rarely, if ever, in practice. In this paper, we consider situations where the available measurements of $x(k)$ are *quantized*. Specifically, let $\Delta_1,\ldots,\Delta_n$ be given positive numbers; define $\mathrm{q}_\Delta : \mathrm{R}^n \to \mathrm{Z}^n$ as follows:

$$\mathrm{q}_\Delta(x) = (j_1,\ldots,j_n)$$

if and only if

$$x \in U_{j_1 \dots j_n} = [(j_1-\tfrac{1}{2})\Delta_1,(j_1+\tfrac{1}{2})\Delta_1) \times \dots \times [(j_n-\tfrac{1}{2})\Delta_n,(j_n+\tfrac{1}{2})\Delta_n)\ .$$

The mapping q is meant to model a *directionwise uniform quantizer* with sensitivities $\Delta_1, \dots, \Delta_n$; we think of $\{q_\Delta(x(k)) : k \geq 0\}$ as the only measurements of the system's state that are available to us, and we require that each $u(k)$ depend solely upon $\{q_\Delta(x(l)) : 0 \leq l \leq k\}$.

In what follows, we investigate the dynamics of the closed loop system obtained from (*) when an instantaneous linear state feedback control law is implemented "naively" by substituting for each $x(k)$ the *center* of the "quantization block" that the measurement $q_\Delta(x(k))$ points out as containing $x(k)$. Such an implementation strategy is standard in practice, where systems are usually designed initially as if no measurement quantization were present. See [2-5] for an extensive discussion of related research, various philosophical issues, and earlier results of a similar nature.

Accordingly, let $y(k)$ be given by

$$[y(k)]_i = \Delta_i [q_\Delta(x(k))]_i\ , \quad 1 \leq i \leq n\ , \quad k \geq 0\ .$$

Let the $(m \times n)$ matrix F be given, and consider the difference equation

$$x(k+1) = Ax(k) - BFy(k)\ , \quad k \geq 0\ . \tag{**}$$

We occupy ourselves throughout the remainder of the paper attempting to ascertain how the behavior of (**) depends on properties A, B, and F. We shall see that the dynamics of (**) turn out to be extraordinarily complicated even in those cases that appear, *prima facie*, to be quite simple.

2. STABILIZATION: NEBULOUS 'EQUILIBRIA' AND INVARIANT MEASURES

Suppose that (A,B) is a reachable (or at least discrete-time stabilizable) pair. Let F be chosen so that all the eigenvalues of $(A-BF)$ lie in the open unit disc $|z| < 1$. It is shown in [5] that there exists a bounded region $\hat{D}_o \subset \mathbf{R}^n$ containing the origin such that every initial condition $x(0) \in \mathbf{R}^n$ for (**) gives rise to a trajectory that enters $\hat{D}_o$ eventually and, once having entered, never leaves. The proof of this assertion is fairly straightforward; it is independent of the quantizer sensitivities $\{\Delta_i\}$, but rests heavily on the fact that q_Δ is uniform in each direction in $\mathbf{R}^n$. If A has at least one eigenvalue whose magnitude exceeds unity, then what happens *inside* the region $\hat{D}_o$ can be extremely complicated.

The fact that $\hat{D}_o$ is invariant under the closed loop dynamics of (**) prompts us to search inside $\hat{D}_o$ for equilibria, periodic solutions, and other simple invariant subsets. In the case $n = 1$, we can paint a fairly complete picture; for general n, we have the following.

2.1 Lemma: *In the notation of the foregoing, suppose that 1 is not an eigenvalue of* A *or* $A-BF$. *The closed loop difference equation (**) has an equilibrium in the set* $U_{j_1 \dots j_n} \subset \mathbf{R}^n$ *if and only if the point* $(j_1\Delta_1, \dots, j_n\Delta_n)$ *lies in the parallelepiped* $\Gamma(U_{0\dots0}) = \{\Gamma x : x \in U_{0\dots0}\}$, *where* $\Gamma = [I-(A-BF)]^{-1}(I-A)$. *Moreover, there is at most one equilibrium of (**) in any* $U_{j_1 \dots j_n}$.

Proof: If $\bar{x} \in \mathbf{R}^n$ is an equilibrium of (**), and $[\bar{y}]_i = \Delta_i[q_\Delta(\bar{x})]_i$, $1 \leq i \leq n$, then we may shuffle (**) to read

$$-[I+(I-A)^{-1}BF]\bar{y} = \bar{x} - \bar{y}\ .$$

The right-hand side of the last identity lies in $U_{0\dots0}$; the left-hand side is just $-\Gamma^{-1}\bar{y}$. Hence if $\bar{x} \in U_{j_1 \dots j_n}$, then $\bar{y} = (j_1\Delta_1, \dots, j_n\Delta_n)$ must lie in $-\Gamma(U_{0\dots0}) = \Gamma(U_{0\dots0})$. Conversely, $\bar{y} = (j_1\Delta_1, \dots, j_n\Delta_n) \in \Gamma(U_{0\dots0})$ implies that $\bar{x} = -(I-A)^{-1}BF\bar{y}$ is an equilibrium of (**) and lies in $U_{j_1 \dots j_n}$. The statement about uniqueness is a consequence of the formula for $\bar{x}$ in terms of $\bar{y}$. □

We remark that Lemma 2.1 holds regarless of where A's eigenvalues are located so long as none of them lies exactly on the unit circle. Consequently, even if the system (*) is zero-input stable, the use of a quantized feedback law as in (**) can cause the closed loop system to have "extra" equilibria. We shall have more to say about this phenomenon in §4; see also [4]. The problem of determining the number and location of the extra equilibria of (**) is quite difficult; it is essentially an exercise in combinatorial geometry, and hinges on many parameters of the problem. For example, the parallelepiped $\Gamma(U_{0\ldots0})$ can have a large area (meaning that $|\det(\Gamma)|$ is large) but still not contain any point of the form $(j_1\Delta_1, \ldots, j_n\Delta_n)$ except zero. Nonetheless, it is easy to bound from above the individual coordinates (and hence the infinity norm) of any extra equilibrium in terms of the 1-norms of the rows of Γ; simply look at where Γ sends the corners of $U_{0\ldots0}$. A fairly short calculation shows that if $\bar{x}$ is an equilibrium of (**), then

$$\|\bar{x}\|_\infty \le \tfrac{1}{2}\|\Gamma\|_\infty\Delta_{\max} ,$$

where $\|\Gamma\|_\infty$ denotes the highest of the 1-norms of the rows of Γ and $\Delta_{\max}$ is $\max\limits_{1\le i\le n}\{\Delta_i\}$. It follows that the *number* of extra equilibria is bounded from above by the number of points of the form $(j_1\Delta_1, \ldots, j_n\Delta_n)$ that lie in the n-cube with sides $\tfrac{1}{2}\|\Gamma\|_\infty\Delta_{\max}$ centered at the origin; this number is at most $(\tfrac{1}{2}\|\Gamma\|_\infty\Delta_{\max}/\Delta_{\min})^n$.

Lemma 2.1 describes the simplest invariant sets for the closed loop system (**); the discovery that (**) usually has equilibria other than the origin gives us only a glimpse at the complexity of its global dynamical behavior. We remarked at the beginning of the section (see also [5]) that if $(A-BF)$ has stable eigenvalues, then there is an invariant region $\hat{D}_o$ for (**) that contains the origin and "attracts" all the trajectories of (**). Inside $\hat{D}_o$, we find a *bona fide* strange attractor for the closed loop system when A has at least one eigenvalue whose magnitude exceeds unity. In that case, every equilibrium $\bar{x}$ of (**) (the origin, for example) lying in the interior of a quantization block $U_{j_1\ldots j_n}$ is unstable because (**) linearizes to $x(k+1)-\bar{x} = A(x(k)-\bar{x})$ near $\bar{x}$. Since every equilibrium of (**) lies in $\hat{D}_o$, we have found a bounded invariant set containing (perhaps many) unstable equilibria; the existence of such a set is practically a *sine qua non* for chaotic behavior. In general, $\hat{D}_o$ also contains many *periodic solutions* for (**); our description in §3 of the case $n = 1$ shows that there are periodic solutions of many different periods under some mild conditions. Each of these periodic solutions is unstable: if we think of (**) as defining $G : \mathbf{R}^n \to \mathbf{R}^n$ via $x(k+1) = G(x(k))$, then each point $\tilde{x}$ on a period-L solution to (**) is a fixed point of G^L, and is unstable because $x(k+L) = G^L(x(k))$ linearizes near $\tilde{x}$ to $x(k+L)-\tilde{x} = A^L(x(k)-\tilde{x})$.

We can regard the region $\hat{D}_o$, or at least the strange attractor it contains, as constituting a sort of *nebulous equilibrium* for (**) that absorbs all the closed loop trajectories. This characterization is by no means as amorphous as it might appear; indeed, the ergodic theory of dynamical systems enables us to make it rigorous. The key idea is that of an *equilibrium measure* for a (probabilistic) dynamical system; roughly speaking, an equilibrium measure is a probability measure on the system's state space that reflects the asymptotic behavior of "almost all" of the system's trajectories. The trajectories of (**), once they enter $\hat{D}_o$, behave in an essentially random manner if A has unstable eigenvalues. This asymptotic pseudorandomness is rooted firmly and quantitatively in the existence of an equilibrium measure concentrated on $\hat{D}_o$. Thus the closed loop system (**), although it is unstable in the traditional sense (i.e. locally near zero), is seen to behave (globally and asymptotically) in a way that can be described as "statistically stable."

Figures 1 and 2 (appended after the References) contain plots obtained from numerical simulations of two 2-dimensional systems of the form (**). Each of Figures 1(a) and 2(a) illustrates the asymptotic distribution of points on typical trajectories of one of the systems. Figure 2(a) displays an equilibrium measure whose support consists of two disjoint subsets of $\mathbf{R}^2$; almost every initial condition for (**) leads to a trajectory that is contained eventually in *one* of them. It is evident that the equilibrium measure corresponding to Figure 2(a) is concentrated on a much "thinner" subset of $\mathbf{R}^2$ than the measure corresponding to Figure 1(a). Unlike Figure 2(a), Figure 1(a) would appear to depict the density of a measure that is absolutely continuous with respect to Lebesgue measure. We obtain Figures 1(b) and 2(b) by performing a "signal-plus-noise" analysis of the same two systems; first rewrite (**) as

$$x(k+1) = (A-BF)x(k)+BFe(k)\ , \quad k \geq 0\ ,$$

where $e(k) = x(k)-y(k)$ for each $k \geq 0$, and then model the $\{e(k)\}$ sequence as componentwise independent white noise distributed uniformly over the "zero block" $U_{0\ldots0}$. Figure 1(b) bears a rough global resemblance to Figure 1(a), but displays none of the fine local structure. Figure 2(b), on the other hand, looks quite unlike Figure 2(a), indicating that a signal-plus-noise analysis of the system (**) corresponding to Figure 2 might be misleading. It is easy to check that no signal-plus-noise analysis will give rise to a limiting distribution that is biased away from the origin in the same way as the equilibrium density in Figure 2(a). Interestingly, it turns out that each of the two "lobes" in Figure 2(a) attracts roughly a half-plane of initial conditions; simulations seem to indicate that the boundary between the "basins" of the two lobes might be fractal!

We summarize the mathematics behind Figures 1 and 2 as follows.

2.2 Theorem: *Given the system (**), suppose that* $|\det(A)| > 1$ *and that the magnitude of every eigenvalue of* $(A-BF)$ *is less than 1. Then there exists on* $\mathbf{R}^n$ *a probability measure* ν_F *with the following properties:*

(a) The support of ν_F *is compact.*

(b) ν_F *is invariant under the dynamics of (**).*

(c) The support of ν_F *breaks up into finitely many disjoint subsets each of which is invariant under the dynamics of (**). Moreover, on each of these subsets, the dynamics of (**) are ergodic with respect to* ν_F.

(d) If every eigenvalue of A has magnitude greater than 1, then ν_F *is absolutely continuous with respect to Lebesgue measure.*

Proof (sketch): [The argument follows along the lines of [8-9], but is far easier because the dynamics of (**) are piecewise linear. The eigenvalue condition in (d) is essential for absolute continuity of ν_F with respect to Lebesgue measure. ☐

The invariant measure ν_F whose existence is guaranteed by Theorem 2.2 is special in the sense that it is *physically meaningful;* we obtain ν_F asymptotically by propagating a normalized restriction of Lebesgue measure through the mapping $x(k) \to x(k+1) = G(x(k))$ defined by (**). The procedure goes as follows. If μ is a probability measure on $\mathbf{R}^n$, let $G_*(\mu)$ be the measure defined by

$$G_*(\mu)(U) = \mu(G^{-1}(U))$$

for every measurable $U \subset \mathbf{R}^n$. It turns out that

$$\mu_N = \frac{1}{N}\sum_{k=0}^{N-1} G_*^k(\mu)$$

converges as $N \to \infty$ to ν_F when μ is taken to be normalized Lebesgue (i.e. uniform) measure on the invariant region region $\hat{D}_o$. In some cases, $G_*^N(\mu)$ also converges to ν_F. The convergence takes place in the weak topology on the space of all probability measures on $\mathbf{R}^n$. We remark that there are usually other measures on $\mathbf{R}^n$ that are preserved by G; a trivial example is the point measure that assigns measure 1 to every measurable subset of $\mathbf{R}^n$ containing the origin and zero to other measurable sets. (Observe that G is vacuously ergodic with respect to this last measure.) ν_F is different because it is a "limiting distribution" that arises naturally when we let the dynamics of the system "operate" on an initial condition that is, at least intuitively, "distributed at random." The finite ergodic decomposition of ν_F is important because it lends some validity to the assertion that "typical" initial conditions for (**) will give rise to trajectories that reflect the equilibrium measure ν_F. If ν_F is *equivalent* to Lebesgue measure, as is the case under certain circumstances when $n = 1$, then we can make an even stronger statement, namely, that "Lebesgue almost every" initial condition yields a trajectory for (**) that is distributed according to ν_F.

Computing the measure ν_F is no easy task. One can obtain excellent approximations by iterating G_*, however, and we have done that in several cases. We are currently investigating the manner in which ν_F and its various statistics (e.g. dispersion and moments) depend on the parameters of the system (**). We have found that there are certain parameter ranges over which ν_F appears to vary "continuously" with respect to A, B, and F; there are, however, places where the character of ν_F seems to "jump" as the system's parameters are changed smoothly. These sudden transitions are reminiscent of bifurcations, since they reflect drastic changes in the limiting qualitative dynamical behavior of (**) as the system's parameters are varied. The case $n = 1$, to which we now turn our attention, is especially illuminating.

3. EQUILIBRIUM MEASURES FOR ONE-DIMENSIONAL SYSTEMS

Consider the system (**) in the case $n = 1$; the matrix A is a real number a, B is a real row m-vector b, and F is a real column m-vector f. For each $k \geq 0$, $y(k) = j\Delta$, where $\Delta > 0$ is given, if and only if $x(k) \in I_j = [(j-\frac{1}{2}\Delta),(j+\frac{1}{2})\Delta)$. The region $\hat{D}_o \subset \mathrm{R}$ becomes an interval centered at zero; for every f, the mapping $x \rightarrow G(x) = ax - bfq_\Delta(x)$ is linear with slope a on each I_j. The reader can check that the conclusion Lemma 2.1 reduces to the following statement: the interval I_j contains an equilibrium of (**) if and only if

$$\frac{-1}{2|j|}|a-1|+1 < a-bf < \frac{1}{2|j|}|a-1| .$$

In §4, we take a closer look at this last relation, which is between the "open loop pole" a of the system (*) and the closed loop pole $a-bf$ of (**); see also [4].

Assume now that $|a| > 1$ and $|a-bf| < 1$, so that the system (**) represents a stabilized version of a system that was originally unstable. It is not too difficult (see [4-5]) to compute the *smallest* interval $[-x^*, x^*]$ about zero that is invariant under the dynamics of (**). This interval does not necessarily contain all the equilibria of (**), but *does* support the equilibrium measure ν_f whose existence is guaranteed by Theorem 2.2. The computation of x^* and the proof of the last assertion are contained in [5]; suffice it to say that x^* grows approximately linearly in the "inverse stability margin" $[1-|a-bf|]^{-1}$.

The existence of the invariant measure ν_f in the case $n = 1$ follows directly from the fundamental results of Li, Lasota, and Yorke on piecewise monotonic mappings of intervals; see [8-9]. The interval mappings that we consider in this paper are particularly simple; not only are they piecewise linear, but they have the same slope wherever their derivative exists. These special properties enable us to compute the measure ν_f exactly in some specific instances. In these cases, ν_f turns out to be piecewise uniform in the sense that the interval $[-x^*, x^*]$ breaks up into a disjoint union of subintervals $\{V_j : 1 \leq j \leq N\}$ over each of which ν_f has a constant density. The computation of ν_f in these special situations hinges upon the notion of a *Lebesgue-Markov partition* for the interval $[-x^*, x^*]$ with respect to the mapping $x \rightarrow G(x)$ defined by (**).

3.1 Definition: A partition $\{V_j : 1 \leq j \leq N\}$ of $[-x^*, x^*]$ is said to be a *Lebesgue-Markov partion with respect to* G if and only if the following condition holds: whenever $G(V_i)$ intersects V_j in a set of positive Lebesgue measure, G is m_{ij}-to-one on $G^{-1}(V_i) \cap V_j$, and $G(G^{-1}(V_i) \cap V_j)$ differs from V_i by a set of Lebesgue measure zero. ☐

Lebesgue-Markov partitions share many properties with Markov partitions for Anosov and Axiom A diffeomorphisms of compact manifolds; see [1] and [7] for details. In the present context, the existence of a Lebesgue-Markov partition enables us to apply some central results from finite-state Markov process theory in computing ν_f.

Let $\{V_j : 1 \leq j \leq N\}$ be a partition of $[-x^*, x^*]$ that is Lebesgue-Markov with respect to G. Denote Lebesgue measure by λ; define the $(N \times N)$ matrix Φ by

$$[\Phi]_{ij} = \frac{\lambda(G^{-1}(U_i) \cap U_j)}{\lambda(U_j)} \ , \quad 1 \le i,j \le N \ .$$

It is evident that the columns of the matrix Φ sum to one; thus, by the Perron-Frobenius Theorem for positive matrices (see [6], for example), there exists an N-vector π whose entries are nonnegative and sum to one such that $\Phi\pi = \pi$. Some genericity assumptions (e.g. irreducibility) make π unique, but for now what is important to us is that π exists. It turns out that $\{V_j\}$'s being a Lebesgue-Markov partition with respect to G implies that ν_f is precisely the piecewise uniform measure ν_ϕ on $[-x^*, x^*]$ whose density ϕ is given by

$$\phi(x) = \frac{\pi_i}{\lambda(V_i)} \ , \quad x \in V_i \ .$$

To see this, let V be a subset of V_i for some i, $1 \le i \le N$. We'll show that $\nu_\phi(V) = \nu_\phi(G^{-1}(V))$, from which it follows readily that G preserves ν_ϕ, and hence that $\nu_f = \nu_\phi$ by the uniqueness results of [8-9]. First of all, note that the Lebesgue-Markov property and the fact that the slope of G is a almost everywhere imply that

$$[\Phi]_{ij} = \frac{\lambda(G^{-1}(V_i) \cap V_j)}{\lambda(V_j)} = \frac{m_{ij}}{|a|}\frac{\lambda(U_i)}{\lambda(U_j)}$$

for every i and j such that $G^{-1}(V_i) \cap V_j$ is nonempty. Furthermore,

$$\nu_\phi(V) = \pi_i \frac{\lambda(V)}{\lambda(V_i)} \ ,$$

and

$$\nu_\phi(G^{-1}(V)) = \sum_j \frac{\lambda(G^{-1}(V) \cap V_j)}{\lambda(V_j)} \ ,$$

where the sum is taken over all j such that $G^{-1}(V) \cap V_j$ is nonempty. The Lebesgue-Markov property of the partition $\{V_j\}$ and the fact that the slope of G is a almost everywhere again enable us to rewrite the sum as follows:

$$\nu_\phi(G^{-1}(V)) = \sum_j \pi_j \frac{m_{ij}}{|a|}\frac{\lambda(V)}{\lambda(V_j)} = \frac{\lambda(V)}{\lambda(U_i)} \sum_j \pi_j \Big[\frac{m_{ij}}{|a|}\frac{\lambda(U_i)}{\lambda(U_j)}\Big] \ .$$

The bracketed quantity is precisely $[\Phi]_{ij}$, and since $\sum_j [\Phi]_{ij}\pi_j = \pi_i$, we get

$$\nu_\phi(G^{-1}(V)) = \pi_i \frac{\lambda(V)}{\lambda(V_i)} = \nu_\phi(V) \ ,$$

and we're done.

It is now possible to justify the "Lebesgue-Markov" terminology of Definition 3.1. Define a random process $\{z(k) : k \ge 0\}$ as follows: let $x(0)$ be distributed uniformly on $[-x^*, x^*]$, and let the random process $k \to x(k)$ evolve according to (**); that is, $x(k) = G^k(x(0))$, $k \ge 0$. Set $z(k) = j$, $1 \le j \le N$, $k \ge 0$, if and only if $x(k) \in V_j$. Thus $\Pr\{z(k) = j\} = \Pr\{x(k) \in V_j\}$ for $1 \le j \le N$ and $k \ge 0$. The fact that $\{V_j\}$ is a Lebesgue-Markov partition guarantees that $\{z(k)\}$ is a Markov chain with stationary transition probabilities and that the distribution of $x(k)$ is, for every $k \ge 0$, uniform over each partition element V_j, $1 \le j \le N$. $\{z(k)\}$ is *not* a stationary Markov chain; the initial distribution (on $z(0)$) is uniform, but the distribution of $z(k)$ approaches π as $k \to \infty$ by the Perron-Frobenius Theorem provided that Φ is aperiodic and positively recurrent. Moreover, the density of $x(k)$ approaches ν_f as $k \to \infty$, as well. The density of $x(k)$ coincides with that of the (piecewise uniform) measure $G_*^k(\lambda)$, with notation as in §2.

Some important questions come immediately to mind. For example, under what circumstances does there exist a partition of $[-x^*, x^*]$ that is Lebesgue-Markov with respect to G? It is not hard to check that if a is an integer and $bf\,\Delta$ is a rational number p/q, then the partition of $[-x^*, x^*]$

into subintervals of the form $V_j = [(j/2q)\Delta,((j+1)/2q)\Delta)$ is a Lebesgue-Markov partition with respect to G. More generally, it happens that a must be an algebraic number in order for a Lebesgue-Markov partition to exist. To see this, recall that if $\{V_j\}$ is a Lebesgue-Markov partition, then for all j we must have

$$\sum_i \frac{\lambda(G^{-1}(V_i) \cap V_j)}{\lambda(V_j)} = 1 = \sum_i \frac{m_{ij}}{|a|} \frac{\lambda(V_i)}{\lambda(V_j)},$$

whereby

$$\sum_i m_{ij} \lambda(V_i) = |a| \lambda(V_j)$$

for every j, from which it follows that $|a|$ is an eigenvalue of an integer matrix, and is therefore algebraic. (Our convention here is that $m_{ij} = 0$ when $G^{-1}(V_i) \cap V_j$ is empty.)

We haven't managed yet to answer several other important questions about Lebesgue-Markov partitions and their relationships with the piecewise uniformity of the special invariant measure ν_f. For one thing, we don't know whether the existence of a Lebesgue-Markov partition is a *necessary* condition for the piecewise uniformity of ν_f. We have no clue as to how big a ''smallest'' Lebesgue-Markov partition must be as a function of the parameters of (**). We don't know, under the assumption that there exists a Lebesgue-Markov partition, whether every partition has a Lebesgue-Markov refinement. All of these questions are currently under investigation.

We certainly cannot expect, in general, that the special invariant measure ν_f will be piecewise uniform. We are prompted to ask whether, and if so in what fashion, the measure ν_f depends *continuously* on a, b, and f. It turns out that this dependence is not globally continuous, as can be seen by looking at the behavior of x^* when the parameters of the system are varied. It is shown in [5] that, for fixed a and b, x^* increases piecewise linearly in $[1-(a-bf)]^{-1}$ as $a-bf$ approaches 1 from below and *jumps* at a countable set of bf-values. Thus, by anyone's definition, ν_f must undergo similar discontinuous transitions as system parameters are changed. It is worth noting that the invariant density for (**) obtained from a signal-plus-noise analysis experiences no such jumps. Furthermore, signal-plus-noise analysis always predicts an invariant measure that assigns positive measure to every neighborhood of zero, whereas the actual invariant measures ν_f for (**) need not do so (compare Figure 2(a)). For instance, let $a = 3/2$ and $bf = 5/8$ in (**); then $x^* = (3/4)\Delta$, and ν_f assigns measure zero to the interval $[-(1/8)\Delta,(1/8)\Delta]$. For more details, see [5].

An interesting feature of the system (**) in the case $n = 1$ is the existence of *Cantor sets* that are invariant under its dynamics. To simplify matters, consider (**) when $a > 1$ and $0 < a-bf < 1$. Recall the criterion at the beginning of this section for the existence of a fixed point for G (i.e., an equilibrium for (**)) in the interval $I_j = [(j-\frac{1}{2}\Delta),(j+\frac{1}{2})\Delta)$; suppose that $\bar{x}_j$ and $\bar{x}_{j+1}$ are fixed points in the adjacent intervals I_j and I_{j+1}, respectively. Let us graph the restriction of the mapping G to the interval $[\bar{x}_j,\bar{x}_{j+1}]$.

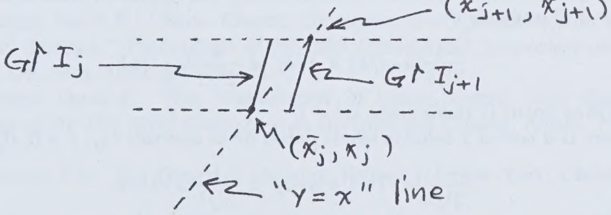


Figure 3

Evidently, a central subinterval of $[\bar{x}_j,\bar{x}_{j+1}]$ gets mapped out of $[\bar{x}_j,\bar{x}_{j+1}]$ in one time step; a central subinterval of each of the two remaining outer subintervals gets mapped out in *two* steps; and so on. We conclude that the set of all initial conditions in $[\bar{x}_j,\bar{x}_{j+1}]$ whose trajectories remain in the interval for all time is a Cantor set, and is invariant under G by construction. The presence of invariant Cantor sets is another commonly encountered piece of evidence for chaotic behavior.

To close the section, we present a general formula that gives the density ϕ_f for the invariant measure ν_f at every point where ϕ_f is defined. For each $x \in [-x^*, x^*]$, let $N_k(x)$ denote the number of inverse images of x under the mapping G^k restricted to $[-x^*, x^*]$. Given a (very small) $\varepsilon > 0$, we can approximate the measure of $[x-\varepsilon, x+\varepsilon]$ by $2\varepsilon\phi_f(x)$. Proceeding formally, write

$$\lambda(G^{-k}([x-\varepsilon, x+\varepsilon])) \sim 2\varepsilon |a|^{-k} N_k(x) ;$$

since $\nu_f = \lim_{k \to \infty} G_*^k(\lambda)$, it follows that $\phi_f(x)$ ought to be given by

$$\phi_f(x) = \lim_{k \to \infty} |a|^{-k} N_k(x) .$$

The last identity can, in fact, be demonstrated rigorously.

4. POLE ASSIGNMENT

In §§2-3, we considered mainly the feedback stabilization of zero-input unstable systems of the form (*). Lemma 2.1 holds also for stable A, however, and prompts us to consider the many dynamical complications that quantized state feedback can introduce through (**) even when the original system is stable. State feedback control of open-loop stable systems occurs frequently in applications; for example, one might apply state feedback to a stable system in order to shift the system's poles to desired locations or to control the system optimally with respect to some cost criterion.

By Lemma 2.1, quantized state feedback can cause the closed loop system (**) to have equilibria other than the origin. These new equilibria are all (locally) asymptotically stable when all the eigenvalues of A lie inside the open unit disc. As it happens, (**) can have nontrivial periodic solutions, as well, and these are also locally asymptotically stable under the same assumption on A. A careful analysis shows that ''most'' initial conditions for (**) lead to trajectories that are attracted to stable periodic solutions or to stable equilibria other than zero when such additional stable limit sets exist. In many cases, the domains of attraction of the various stable limit sets are disconnected and impinge upon each other in a complicated fashion reminiscent of the fractal boundary between the basins of the two ''lobes'' of the equilibrium measure depicted in Figure 2(a). The end result is that quantized state feedback can lead to a multiplicity of stable limiting behaviors for (**), most of which are biased away from zero.

A quick discussion of the case $n = 1$ is informative; a more detailed treatment appears in [4], in which the following result is proven.

4.1 Theorem: Let $n = 1$ in (**). Let $I_j = [(j-\tfrac{1}{2})\Delta, (j+\tfrac{1}{2})\Delta)$. Suppose $|a| < 1$ and $|a-bf| < 1$.

(1) *The system (**) always has a fixed point at zero; moreover, there is a fixed point in the interval* I_j, $j \neq 0$, *if and only if*

$$\frac{-1}{2|j|}|a-1|+1 < a-bf < \frac{1}{2|j|}|a-1|+1 .$$

Each of these fixed points is stable since $|a| < 1$.

(2) *There is a period 2 solution whose points lie in intervals* $I_{\pm j}$, $j \neq 0$, if and only if

$$\frac{-1}{2|j|}|a+1|-1 < a-bf < \frac{1}{2|j|}|a+1|-1 .$$

Each of these periodic solutions is stable since $|a| < 1$.

(3) *Every trajectory of* $k \to x(k)$ *of (**) tends to one of the fixed points or periodic solutions as* $k \to \infty$. □

Theorem 4.1 allows us to draw a root locus plot of sorts for one-dimensional systems of the form (**); the picture resembles a bifurcation diagram, since it indicates parameter ranges that correspond to the different possible qualitative asymptotic behaviors of the system (**).

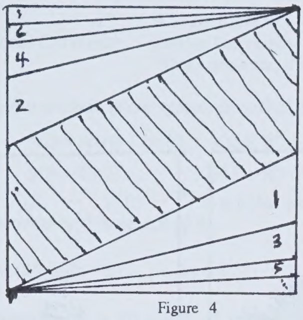


Figure 4

Figure 4 depicts a division of the square $|a| < 1$, $|a-bf| < 1$ into subregions. If $(a, a-bf)$ lies in the central shaded zone, then every trajectory $k \to x(k)$ of (**) approaches zero as $k \to \infty$. For $(a, a-bf)$ in the regions labeled $2j$, there exist $2j$ stable equilibria of (**) other than zero, and and every initial condition $x(0)$ for (**) leads to a trajectory that approaches the equilibrium closest to $x(0)$. If $(a, a-bf)$ falls in the region $2j+1$, (**) has $2j$ stable period-2 solutions, and every $x(0)$ gives rise to a trajectory for (**) that approaches the periodic solution nearest to $x(0)$.

ACKNOWLEDGEMENTS

This research was supported in part by NSF under Grant No. ECS-8352211, by the U.S. Army Research Office through the Mathematical Sciences Institute at Cornell University, by the AFOSR under Grant No. FQ8671-8800448, and by the School of Electrical Engineering, Cornell University.

REFERENCES

[1] Bowen, Rufus R. *Equilibrium States and the Ergodic Theory of Anosov Diffeomorphisms.* New York: Springer-Verlag, Inc., 1975.

[2] Delchamps, David F., ''Controlling the Flow of Information in Feedback Systems With Quantized Measurements,'' to be presented at the 28th IEEE Conference on Decision and Control, Tampa, FL, December, 1989.

[3] Delchamps, David F., ''New Techniques for Analyzing the Effect of Output Quantization in Feedback Systems,'' *Proceedings of the 1988 Conference on Information Sciences and Systems,* Princeton, NJ, March, 1988, pp. 228-232.

[4] Delchamps, David F., ''Some Chaotic Consequences of Quantization in Digital Filters and Digital Control Systems,'' *Proceedings of the 1989 International Symposium on Circuits and Systems,* Portland, OR, May, 1989, pp. 602-605.

[5] Delchamps, David F., ''The 'Stabilization' of Linear Systems With Quantized Feedback,'' *Proceedings of the 27th IEEE Conference on Decision and Control,* Austin, TX, December, 1988, pp. 405-410.

[6] Gantmacher, F.R. *The Theory of Matrices, Volume II.* New York: Chelsea Publishing Co., 1959.

[7] Guckenheimer, J. and P. J. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* New York: Springer-Verlag, Inc., 1983.

[8] Lasota, A. and James A. Yorke, ''On the Existence of Invariant Measures for Piecewise Monotonic Transformations,'' *Transactions of the American Mathematical Society,* Vol. 186 (1973), pp. 481-488.

[9] Li, Tien-Yien, and James A. Yorke, ''Ergodic Transformations From an Interval Into

Itself,'' *Transactions of the American Mathematical Society*, Volume 235 (1978), pp. 183-192.

[10] Walters, Peter. *An Introduction to Ergodic Theory.* New York: Springer-Verlag, Inc., 1982.

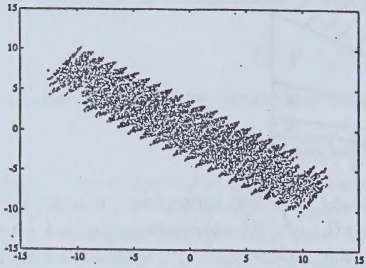


Figure 1(a)

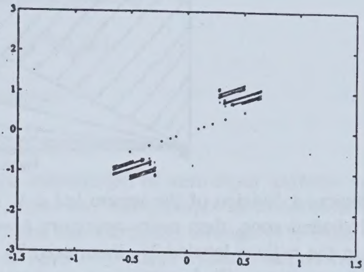


Figure 2(a)

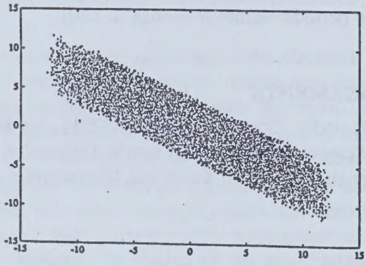


Figure 1(b)

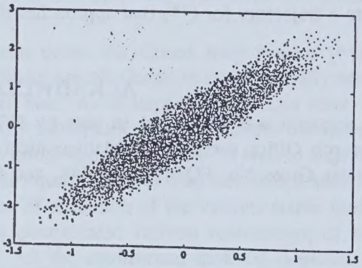


Figure 2(b)

MOMENT MATCHING MODEL REDUCTION FOR MULTIRATE LINEAR SYSTEMS

D. WILLIAMSON
The Austrailian National University
Canberra, Australia

R.E. SKELTON
Purdue University
West Lafayette, IN 47907

G. ZHU
Purdue University
West Lafayette, IN 47907

Consider a continuous time linear system described by

$$\begin{aligned} \dot{x}(t) &= \tilde{A}x(t) + \tilde{B}u(t) \ ; \ x(t) \in R^{n_x}, \ u(t) \in R^{n_u} \\ y(t) &= Cx(t) \qquad\qquad\quad ; \ y(t) \in R^{n_y} \end{aligned} \tag{1a}$$

where $u^*(t) = [u_1^*(t), u_2^*(t), \cdots, u_j^*(t) \cdots]$ is a pulse amplitude modulated (PAM) signal in which the j^th^ block of components $u_j(t)$ is described by

$$u_j(t) = \sum_k u_j(kT_j)p_j(t - kT_j) \ ; \ u_j(t) \in R^{n_{uj}} \qquad p_j(\sigma) = \begin{cases} 1 \ ; \ 0 \le \sigma < T_j \\ 0 \ ; \ \text{otherwise}. \end{cases} \tag{1b}$$

The mth block of components $y_m(t) \in R^{n_{yj}}$ of the output $y^*(t) = [y_1^*(t), y_2^*(t) \cdots, y_m^*(t) \cdots]$ is assumed to be sampled every S_m seconds.

This multirate sampling strategy may be employed when different components of the continuous time system (1) exhibit different characteristic frequencies, and so demand different minimum sampling rates. In addition, different input modulation rates $\{T_j^{-1}\}$ may be needed depending on the speed and hardware constraints as a result of the computational requirements for realizing the digital control algorithm. We assume $\{T_j^{-1}\}$ and $\{S_m^{-1}\}$ are all rationally related in which case (1) describes a periodic time-varying system.

Model reduction methods are employed to approximate the response of a system to certain classes of input signals by another system of lower dimension. For the PAM input signal (1b) we shall show that the response $\{y_m(kS_m) \ ; \ k \text{ integer}\}$ at the measurement instants of (1a) can be expressed exactly in terms of a discrete system with discrete inputs $\{u_j(kT_j) \ ; \ k \text{ integer}\}$. It therefore makes sense in this case to base any model reduction technique on such a discrete system and the specific modulation rates $\{T_j^{-1}\}$ and sampling rates $\{S_m^{-1}\}$ rather than on the continuous time system (1a) in which the PAM nature of u(t) is not explicitly used.

Assume all input rates are the same so that in (1b)

$$T_j = T \triangleq p\Delta \tag{2a}$$

and all measurement rates are the same so that

$$S_m = S \triangleq r\Delta \tag{2b}$$

where LCM (p, r) = 1 is the lowest common multiple of p and r. Hence, in a time of $rp\Delta$ seconds, all components of the input amplitude u(t) are changed at most r times, and all components of the output y(t) are measured exactly p times.

Then for all integers k, the state $x(k) \triangleq x(t = k\Delta)$ is given precisely in terms of $u(k) \triangleq u(t = k\Delta)$ in (1.1) by

Lemma 1

The system (1) *in which all* n_u *components of the input vector* u(t) *are only updated every* $p\Delta$ *seconds, and all* n_y *components of the measurement vector* y(t) *are only recorded every* $r\Delta$ *seconds is described for* $\kappa = 0, 1, 2, \dots$ *by*

$$\begin{aligned} z(\kappa + 1) &= Fz(\kappa) + Gv(\kappa) \\ w(\kappa) &= Mz(\kappa) + H_0 v(\kappa) \end{aligned} \tag{3a}$$

where

$$z(\kappa) \in R^{n_x} \ , \ v(\kappa) \in R^{rn_u} \ , \ w(\kappa) \in R^{pn_y}$$

and

$$\begin{aligned} F &\triangleq A^{pr} \ ; \ G = [B_1, B_2, \dots, B_r] \\ M &\triangleq [C^*, (CA^r)^*, \cdots, (CA^{(p-1)r})^*]^* \\ H_o &\triangleq [M_{ij}] \end{aligned} \tag{3b}$$

where the $n_y \times n_u$ *matrix* M_{ij} *and where the* $n_x \times n_u$ *matrix* B_j *is given by*

$$B_j = A^{p(r-j)} \sum_{i=0}^{p-1} A^i B$$

$$M_{ij} = \begin{cases} \sum_{n=1}^{Q_{ij}} CA^{(i-1)r-(j-1)p-n}B & ; \ (i-1)r > (j-1)p \\ 0 & ; \ \text{otherwise} \end{cases} \tag{4a}$$

with

$$Q_{ij} \triangleq \min\{p, (i-1)r-(j-1)p\} \ . \tag{4b}$$

The input $v(\kappa)$ *and output* $w(\kappa)$ *are defined by*

$$\begin{aligned} v^*(\kappa) &= [u^*(\kappa pr), u^*(\kappa pr+p), \ \cdots, u^*(\kappa pr + rp - p)] \\ w^*(\kappa) &= [y^*(\kappa pr), y^*(\kappa pr + r), \ \cdots, y^*(\kappa pr + rp - r)] \ . \end{aligned} \tag{4c}$$

Lemma 2

The system (1) in which for j = 1, 2 the n_{uj} *components of the input vector* $u(t)$ *where* $(n_u = n_{u1} + n_{u2})$ *are updated every* $p_j\Delta$ *seconds, and the* n_{yj} *components of the measurement vector* $y(t)$ *(where* $n_y = n_{y1} + n_{y2}$*) are only recorded every* $r_j\Delta$ *seconds is described for* $\kappa = 0, 1, 2, \ldots.$ *by*

$$\begin{aligned} z(\kappa + 1) &= Fz(\kappa) + \sum_{m=1}^{2} G_m v_m(\kappa) \\ w_l(\kappa) &= M_l z(\kappa) + \sum_{m=1}^{2} \overset{(l)}{H}_{0,m} v_m(\kappa) \ ; \ l = 1, 2 \end{aligned} \tag{5}$$

where for r, p defined by

$$p \triangleq LCM(p_1, p_2) \ ; \ r \triangleq LCM(r_1, r_2) \ ; \ d \triangleq LCM(p, r) \tag{6}$$

$$z(\kappa) \in R^{n_x} , \ v_m(\kappa) \in R^{r n_{um}} \ ; \ w_j(\kappa) \in R^{p n_{yj}}$$

and

$$\begin{aligned} &F = A^d \\ &G_1 = [B_{11}, B_{12}, \ldots, B_{1,d/p_1}] \ ; \ G_2 = [B_{21}, B_{22}, \ldots, B_{2,d/p_2}] \\ &M_l = [C_l^*, (C_l A^{r_l})^*, \ldots, (C_l A^{d-r_l})^*]^* \ ; \ l = 1, 2 \\ &\overset{(l)}{H}_{0,m} = [\overset{(lm)}{M}_{ij}]. \end{aligned} \tag{7}$$

The $n_x \times n_{uj}$ *matrices* B_{jm} *and the* $n_{yl} \times n_{um}$ *matrices* $\overset{(lm)}{M}_{ij}$ *are given by*

$$B_{jm} = A^{d-mp_j} \sum_{\alpha=0}^{p_j-1} A^{\alpha} B_j \tag{8}$$

the $n_{yl} \times n_{um}$ *block matrix* $\overset{(lm)}{M}_{ij}$ *is given by*

$$\overset{(lm)}{M}_{ij} = \begin{cases} \sum_{n=1}^{Q_{ijlm}} C_l A^{(i-1)r_l-(j-1)p_m-n} B_m & ; \ (i-1)r_l > (j-1)p_m \\ 0 & ; \ \text{otherwise} \end{cases} \tag{9}$$

where

$$Q_{ijlm} = \min\{p_m, (i-1)r_l - (j-1)p_m\}$$

The input vectors $v_l(\kappa)$ *and output vectors* $w_l(\kappa)$ *are defined for* $l = 1, 2$ *by*

$$\begin{aligned} v_l^*(\kappa) &= [u_l^*(\kappa d), u_l^*(\kappa d + p_l), \ldots u_l^*(\kappa d + d - p_l)] \\ w_l^*(\kappa) &= [y_l^*(\kappa d), y_l^*(\kappa d + r_l), \ldots, y_l^*(\kappa d + d - r_l)]. \end{aligned} \tag{10}$$

Conclusion.

There are two ways to obtain reduced order models for multirate systems. One may use the full order single-rate model to get reduced order models by some kind model reduction method. Then one may obtain the reduced order multirate model from the reduced order single-rate model. Alternately, one may derive a reduced order multirate model from the full order multirate model directly. For our example the latter method is much better than the former one.

ANALYSIS AND RECOVERY OF SAMPLE-AND-HOLD SIGNAL WITH IRREGULAR SAMPLES

FAROKH MARVASTI
ECE Dept., IIT, Chicago, IL 60616

I. *Introduction*

Digital-to-analog converters are based on sample-and-hold and low pass filtering; this process creates some distortion at the Nyquist rate. To alleviate this problem, a modular method of the recovery of a signal from its sampled-and-held version was described by the author in [1]. It is our intention to generalize this method to sampled-and-held version of nonuniform samples. For the general theorems related to this area, see the author's monograph [2].

II. *Statement of the Problem and Its Solutions*

Assume $x(t)$ to be a bandlimited signal. Let us take the sample-and hold signal $(s(t))$ of a set of nonuniform samples $\{x(t_k)\}$ as shown in Fig. 1. Let us take a 1-1 coordinate transformation such as the one used by Papoulis [3], i.e.,

$$t = \tau - \theta(\tau) \tag{1}$$

such that when $\tau = nT \rightarrow t = t_n$.

$\theta(\tau)$ in (1) could be any nonunique function such that $\theta(nT) = \mu_n = t_n - nT$ is the deviation of nonuniform samples from a set of uniform samples. We can show that a necessary condition for this 1-1 mapping is that $|\mu_n| < \frac{T}{\pi}$. This coordinate transformation maps $x(t)$ into a new function $g(\tau)$ such that

$$x(t) = x(\tau - \theta(\tau)) = g(\tau) \qquad \text{and} \qquad x(t_n) = g(nT) \tag{2}$$

Assuming that the sample-and-hold signal $s(t)$ is mapped into $s_1(\tau)$, we have [1], [2], [4]

$$\sum_{i=-N}^{N} S_1\,(f - iT) \approx G(f), \ |f| < W \tag{3}$$

In τ domain, we have

$$h(\tau) * \left\{ s_1(\tau) \left[1 + 2 \sum_{i=1}^{N} \cos \frac{2\pi i \tau}{T} \right] \right\} \approx g(\tau) \tag{4}$$

where $h(\tau)$ is an ideal low pass filter.

Taking

$$\tau = \gamma(t) = t - \phi(t) \tag{5}$$

where $\phi(t)$ is a bounded function. Now, substituting (5) in (4), and noting that $s_1[\gamma(t)] = s(t)$ and $g[\gamma(t)] = x(t)$, and $h(\gamma(t)) = sinc[\frac{t-\phi(t)}{T}]$, we get

$$sinc[\frac{t-\phi(t)}{T}] * \left\{ s(t) \left[1 + 2 \sum_{i=1}^{N} \cos[\frac{2\pi i t}{T} - \frac{2\pi i}{T}\phi(t)] \right] \right\} \approx x(t) \tag{6}$$

Fig. 2 is the modular representation of equation (6).

III. *Conclusions*

We extended the modular technique that was developed for the recovery of uniform sample-and-hold to the nonuniform case $(|\mu_n| < \frac{T}{\pi})$. Besides the sample-and-hold case, this technique, assuming Eqn. (4) holds, can be generalized to any interpolation (e.g., nth order-hold).

References

[1] F. Marvasti, "A New Method to Compensate for the Sample-and-Hold Distortion," IEEE Trans. ASSP, vol. ASSP-33, no. 3, June 1985.

[2] F. Marvasti, *A Unified Approach to Zero-Crossings and Nonuniform Sampling of Single and Multidimensional Signals and Systems,* NONUNIFORM Publication, (P.O. Box 1505, Oak Park, IL 60304), 1987.

[3] A. Papoulis, "Error Analysis in Sampling Theory," Proc. IEEE, vol. 54, no. 7, pp. 947-955, July 1966.

[4] F. Marvasti, "Reflections on the Poisson Sum Formula," Trans. of Inst. of Elect. and Comm. Eng. of Japan, Section E, vol. E67. no. 9, Sept. 1984.

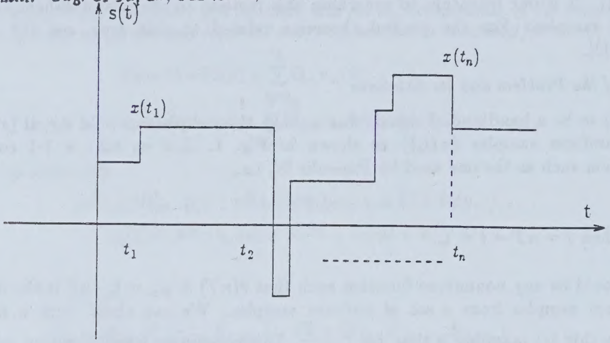


Fig. 1 Sample-and-hold with variable width (of nonuniform samples).

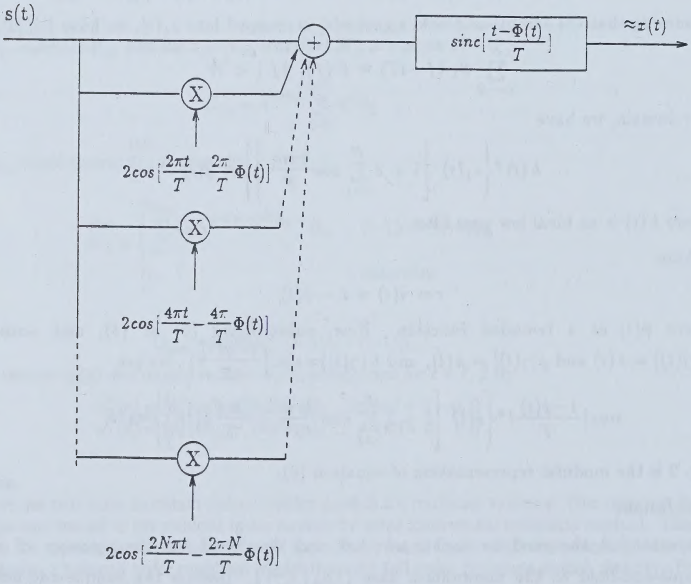

s(t)
+
$sinc[\frac{t-\Phi(t)}{T}]$
$\approx x(t)$
X
$2cos[\frac{2\pi t}{T}-\frac{2\pi}{T}\Phi(t)]$
X
$2cos[\frac{4\pi t}{T}-\frac{4\pi}{T}\Phi(t)]$
X
$2cos[\frac{2N\pi t}{T}-\frac{2\pi N}{T}\Phi(t)]$

Fig. 2

SPECTRAL FILTERING ANALYSIS OF THE CONJUGATE GRADIENT METHOD APPLIED TO SIGNAL EXTRAPOLATION

BARRY J. SULLIVAN[1]
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, Illinois 60208-3118

INTRODUCTION

Iterative search algorithms are frequently applied to ill-conditioned problems in signal processing. It is often necessary to terminate the iteration when numerical errors due to the poor condition of the problem appear to dominate the solution, rather than when some convergence condition is met. Thus it is helpful to monitor the iteration as it progresses in a manner that gives an indication of the quality of the computed solution without prior knowledge of the true solution. The purpose of this paper is to present an analysis framework that permits such monitoring. It is based on the concept of spectral filtering analysis, which is discussed below. We demonstrate the utility of our technique by examining the application of the conjugate gradient method to the signal extrapolation problem.

SPECTRAL FILTERING ANALYSIS

Spectral filtering analysis relates the solution computed by a particular method to the solution space as described by the singular value decomposition (SVD) of the system matrix. To illustrate the concepts of spectral filtering, consider solving the $n \times n$ linear system $Ax = b$, where the rank of A is ρ. The SVD of A can be written as $A = \sum_{i=1}^{\rho} \sigma_i u_i v_i^T$, with the scalars σ_i being the singular values, and the n-vectors u_i and v_i being the left and right singular vectors, respectively [1]. The singular values are assumed to be in nonincreasing order. The pseudoinverse solution can be expressed in terms of the SVD as

$$\hat{x} = A^{\dagger} b = \sum_{i=1}^{\rho} \frac{u_i^T b}{\sigma_i} v_i. \tag{1}$$

When A is full rank ($\rho = n$), $A^{\dagger}$ is equal to A^{-1}.

An ill-conditioned linear system is characterized by a matrix having a spectrum of singular values spanning several orders of magnitude. Clearly, the presence of relatively small singular values can lead to instability in the computation of $\hat{x}$. Spectral filtering methods introduce weighting factors to deemphasize components of the solution associated with small singular values. The spectral filtering solution is

$$\tilde{x} = \sum_{i=1}^{\rho} s(\sigma_i) \frac{u_i^T b}{\sigma_i} v_i. \tag{2}$$

In general, the spectral filter function $s(\sigma_i)$ should tend to unity for large values of σ_i and zero for small values. We have presented a spectral filtering description of the steepest descent method in an earlier paper [2]. In this paper, we outline how a set of spectral filter functions is generated as the conjugate gradient iteration progresses.

A major benefit of expressing the computed results in terms of spectral filter functions is the ability to evaluate the error without prior knowledge of the true solution. Let

$$x_{k+1} = \sum_{i=1}^{\rho} s_{k+1}(\sigma_i) \frac{u_i^T b}{\sigma_i} v_i \tag{3}$$

be the computed solution after $k+1$ iterations of some iterative method, with $s_{k+1}(\sigma_i)$ representing the corresponding spectral filter function. If A is full rank, then $\hat{x}$ as expressed in Equation (1) is identical to the true solution. Suppose instead of the noiseless vector b we observe $\tilde{b} = b + e$, where the n-vector e represents measurement errors. In addition, assume that the elements of e are samples from a zero-mean white noise source with variance ϵ^2, uncorrelated with the elements of b. If b is modeled as uncorrelated samples from a random process with variance δ^2, it is relatively straightforward to show that the mean-square error after $k+1$ iterations is

$$E\{\|x - \tilde{x}_{k+1}\|_2^2\} = \sum_{i=1}^{n} \left[(1 - s_{k+1}(\sigma_i))^2 \frac{\delta^2}{\sigma_i^2} + s_{k+1}^2(\sigma_i) \frac{\epsilon^2}{\sigma_i^2} \right]. \tag{4}$$

---

[1] This research was supported in part by a grant from the Ameritech Foundation.

THE CONJUGATE GRADIENT METHOD AND SIGNAL EXTRAPOLATION

The conjugate gradient method solves the positive definite symmetric system $Ax = b$, by minimizing the function $f(x) = \frac{1}{2}x^T Ax - x^T b$ in an iterative fashion [1]. The method is essentially a sequence of searches along a set of conjugate directions, generated as the iteration progresses, and eventually spanning the solution space. Theoretically, it will converge in no more than n steps for an $n \times n$ system.

Iterative methods like conjugate gradients are typically described by the expressions for the solution vector and various other vectors and scalars generated at each iteration. It is also possible to express the solution at a given iteration in terms of a matrix polynomial. Initializing the iteration with $x_0 = 0$, the polynomial form of the conjugate gradient iteration becomes

$$x_{k+1} = P_k(A)b \tag{5}$$

where $P_k \in \Pi_k = \{\text{polynomials of degree} \leq k\}$. The polynomials P_k can be found via the recursions $(P_{-1} = 0;\ Q_0 = 1)$ [3]

$$\begin{aligned} P_k(\lambda) &= P_{k-1}(\lambda) + \alpha_k Q_k(\lambda), \\ Q_{k+1}(\lambda) &= [1 - \lambda P_k(\lambda)] + \beta_k Q_k(\lambda). \end{aligned}$$

The development of spectral filter functions relies on evaluating the matrix polynomials appearing in (5) in terms of the SVD of A. Since A is positive definite and symmetric, its SVD can be written as $A = U\Sigma U^T$, where the columns of the $n \times n$ matrix U are the singular vectors of A and Σ is a diagonal matrix containing the singular values of A. Any polynomial function of A can be evaluated as [1]

$$\begin{aligned} P(A) &= U\mathrm{diag}[P(\sigma_1), P(\sigma_2), \ldots, P(\sigma_n)]U^T \\ &= \sum_{i=1}^{n} P(\sigma_i) u_i u_i^T. \end{aligned}$$

The polynomial version of the conjugate gradient iteration (5) thus becomes

$$\begin{aligned} x_{k+1} &= \sum_{i=1}^{n} P_k(\sigma_i)(u_i^T p) u_i \\ &= \sum_{i=1}^{n} \sigma_i P_k(\sigma_i) \frac{u_i^T p}{\sigma_i} v_i. \end{aligned} \tag{6}$$

Comparing (6) to (2), it is evident that $\sigma_i P_k(\sigma_i)$ serves as the spectral filter function for the $k + 1$-st conjugate gradient iteration.

The extrapolation of a discrete-time band-limited signal can be solved based on a positive-definite system $Ax = b$ [4]. We use the extrapolation problem to illustrate the utility of spectral filtering analysis. By examining the spectral filter function after each iteration, we evaluate the performance of the conjugate gradient method as applied to this problem. In particular, we show that the the optimal stopping point for the iteration can be determined by monitoring (4). We also use spectral filtering analysis to help explain certain aspects of the conjugate gradient method's performance, such as the effect of changes in sampling rate.

REFERENCES

[1] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore: The Johns Hopkins University Press, 1983.

[2] B. J. Sullivan, "Spectral filtering analysis of descent methods for adaptive filtering," *Proc. 1989 Conf. Info. Sci. and Syst.*, Baltimore, MD, March 1989.

[3] C. R. Vogel, "Solving ill-conditioned linear systems using the conjugate gradient method," Department of Mathematical Sciences, Montana State University, Bozeman, MT 59717.

[4] B. J. Sullivan and B. Liu, "On the use of singular value decomposition and decimation in discrete-time band-limited signal extrapolation," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-32, no. 6, pp. 1201–1212, Dec. 1984.

THE CLASSIFICATION AND ASSOCIATIVE MEMORY CAPABILITY OF STACK FILTERS

PAO-TA YU AND EDWARD J. COYLE
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

The notion of the *on−set* of a positive Boolean function is used to classify stack filters into four different types, called *type-0* through *type-3*.

The associative memory of a stack filter is defined to be the set of root signals of that filter. If the root sets of two stack filters both contain a desired set of patterns, but one filter's root set is smaller than the other, then this filter is said to be *better* for that set of patterns. Some learning schemes are proposed to find *better* stack filters and it is shown that there is always a type-3 stack filter which is better than any type-1 and type-2 stack filter with respect to preserving a desired set of patterns.

INTRODUCTION

Stack filters are a class of nonlinear filters that possess both the threshold decomposition property and the ordering property called the stacking property. In [1,2] it is stated that each stack filter is based on a positive Boolean function which defines a binary filter on each level of the threshold decomposition architecture.

Once the operation of a stack filter on binary signals is understood, its operation on multi-valued signals can be understood through the stacking property. This fact is used throughout this paper; most results will be derived for binary signals only, and then they can be easily extended to multi-valued signals.

By analogy with the median filter, the *root set* of a stack filter is the set of all the signals which are invariant under filtering by that stack filter [1-4]. The root set of a stack filter can be considered to be the *associative memory* of that stack filter. The associative memory of a median filter of window width $2N+1$ is, as shown in [3,4], the set of all signals in which increasing regions are separated from decreasing regions by constant-valued regions of length greater than N.

Our goal in this paper is to study the associative memory capabilities of stack filters. The first step in this study is the classification of stack filters into four different types, called *type-0* through *type-3*. The notions of the *on-set* and *off-set* of the binary filter which defines a stack filter are the primary tools used to develop this classification scheme.

The goal of this study is to determine a stack filter which is *best* in the sense that its root-set is the smallest possible among all stack filters whose root-sets contain a prespecified set of signals or patterns. Such a stack filter is then said to have the smallest number of false memories [5] of all stack filters with the specified set of memories.

Some of the results obtained include a proof that a stack filter which is *best* can always be found in the set of type-3 stack filters. Learning schemes for type-1 through type-3 stack filters are introduced and analyzed. Many examples are given to illustrate the theory and the learning algorithms.

In Section 2, the definitions, notation, and basic concepts required in this paper are introduced. In Section 3, the classification scheme for stack filters and their equivalent Boolean expressions is presented. One decomposition property of type-3 stack filters is also given in this section.

In Section 4, the concept of a *root−preserving lemma* is emphasized. In Section 5, the root-preserving properties and the associative memory capability of stack filters are investigated and the notion of a "better" stack filter is made precise.

BASIC CONCEPTS

In this section, we first review the definition of binary stack filters and some of their properties. Then, the use of *cubic expressions* to represent Boolean functions is introduced. Finally, two useful operators are defined and some of their properties are stated.

Binary Stack Filters

The appended version of a finite binary signal $X = (x_1, x_2, \ldots, x_L)$ with some positive integer N is called X' and is defined as follows:

$$X' = (x'_{-N+1}, x'_{-N+2}, \cdots, x'_{L+N})$$

in which

$$x'_i = x_1, \ -N+1 \leq i \leq 0;$$
$$x'_i = x_i, \ 1 \leq i \leq L;$$
$$x'_i = x_L, \ L+1 \leq i \leq L+N.$$

The binary output signal Y corresponding to the binary input signal X is defined as follows:

$$y_i = f\,(x'_{i-N}, x'_{i-N+1}, \ldots, x'_i, \ldots, x'_{i+N}) \qquad i=1,\ldots,L. \tag{2.1}$$

Note that x'_i is the *ith* component of the appended signal of X and y_i is the *i*th component of the output signal Y from the binary filter.

From (2.1) it is clear that this sliding window filter can be represented by a mapping of the form $S_f(X) = Y$; that is, $S_f\!: \{0, 1\}^L \rightarrow \{0, 1\}^L$. Note that $S_f(\cdot)$ will be used to denote the operation of a binary stack filter based on a positive Boolean function f.

Cubic Expressions

Cubic expressions can be used to explain Boolean functions in terms of geometry or sets in n-dimensional space. As will be seen later, this geometrical interpretation is particularly helpful in the study of stack filters.

The Cartesian product of n copies of β_2 is called the $n-$*cube*, where $\beta_2 = \{0, 1\}$ and n is a positive integer. There are 2^n elements in the $n-$*cube*, and each one is called a vertex.

With the notion of vertices and $n-$*cube*s, subcubes of the $n-$*cube* can be defined. First, a vertex will be called a $0-$*cube* since it is a single point and therefore dimensionless. A $1-$*cube* is a subset of the $n-$*cube* obtained by replacing any one component of any vertex by "x", where "x" means "0" or "1". For example, $0x11 = \{0011, 0111\}$ is a $1-$*cube* and can be thought of as a line between two vertices. A $2-$*cube* is a subset of $n-$*cube* replacing any two components of any vertex by x's. For example, $0x1x = \{0010, 0011, 0110, 0111\}$ is a $2-$*cube*, and can be thought of as a square. Continuing on, an $r-$*cube* is a subset of the $n-$*cube* obtained by replacing r components of any vertex by x's.

There is a one-to-one correspondence between $r-$*cubes* and $(n-r)$-literal fundamental conjunctions (a simple Boolean product of $n-r$ distinct binary variables) where literal means a variable x_i or its complement $\overline{x_i}$. We use several examples to illustrate this one-to-one correspondence:

$$011 \leftrightarrow \overline{x_1} x_2 x_3 \qquad (in\ \beta_2^3) \qquad (0-cube)$$

$01x \leftrightarrow \overline{x_1}x_2 \qquad (in\ \beta_2^3) \qquad (1-cube)$

$01xx0 \leftrightarrow \overline{x_1}x_2\overline{x_5} \qquad (in\ \beta_2^5) \qquad (2-cube)$

$1xx0x \leftrightarrow x_1\overline{x_4} \qquad (in\ \beta_2^5) \qquad (3-cube)$

The correspondence between Boolean expressions and cubic expression, then, can be summarized as follows:

Theorem 2.1:

Let f be a nontrivial Boolean function of n variables. The sum-of-products expression of f can be expressed as

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{k} m_i$$

if and only if the *on−set* of f can be expressed as

$$on(f) = \bigcup_{i=1}^{k} M_i$$

where m_i is a fundamental conjunction, M_i is the corresponding *r−cube* of m_i, k is an appropriate positive integer, $\sum$ is the Boolean "+", and $\bigcup$ is the set union.

□

Note that the notion of an *on−set* and an *off−set* of a Boolean function f of n variables, denoted as $on(f)$ and $off(f)$, respectively; are exactly defined as follows:

$$on(f) = \{ v : v \in \{0, 1\}^n, f(v) = 1 \}$$

and

$$off(f) = \{ v : v \in \{0, 1\}^n, f(v) = 0 \}$$

Positive Extension and Negative Extension Operators

Two powerful operators called the positive extension operator and the negative extension operator are introduced. These operators provide an easy and fast way to deal with the problems associated with stack filters.

Let $\mathcal{P}$ be the set of all subsets of β_2^n; that is, $\mathcal{P}$ is the power set of β_2^n. Two operators from $\mathcal{P}$ to $\mathcal{P}$ are defined as follows:

Definition 2.1:

Let $P\_EXT(\cdot)$ be a function mapping from $\mathcal{P}$ to $\mathcal{P}$. It satisfies the following conditions:

(a) $P\_EXT(\emptyset) = \emptyset$, where $\emptyset$ is the empty set.

(b) If v is vertex in β_2^n, then $P\_EXT(\{v\})$ is the *r−cube* replacing all the 0's components of v by x's, where r is the number of 0's in v.

(c) If $V \in \mathcal{P}$ and V is nonempty, then

$$P\_EXT(V) = \bigcup_{v \in V} P\_EXT(\{v\}).$$

□

Definition 2.2:

Let $N\_EXT(\cdot)$ be a function mapping from $\mathcal{P}$ to $\mathcal{P}$. It satisfies the following conditions:

(a) $N\_EXT(\emptyset) = \emptyset$, where $\emptyset$ is the empty set.

(b) If v is vertex in β_2^n, then $N\_EXT(\{v\})$ is the *r−cube* replacing all the 1's components of v by x's, where r is the number of 1's in v.

(c) If $V \in \widetilde{\mathcal{P}}$ and V is nonempty, then

$$N\_EXT(V) = \bigcup_{v \in V} N\_EXT(\{v\}).$$

□

If the stack filter $S_f(\cdot)$ based on the positive Boolean function f whose *on−set* is $P\_EXT(\{v\})$ for some vertex v, then $S_f(\cdot)$ is a minimal operator or an erosion operator in morphology [6]. Similarly, if the stack filter $S_f(\cdot)$ based on the positive Boolean f whose *off−set* is $N\_EXT(\{v\})$ for some vertex v, then $S_f(\cdot)$ is a maximal operator or an dilation operator in morphology [6]. Note that the definitions of $P\_EXT(\cdot)$ and $N\_EXT(\cdot)$ can be easily extended to high dimension. $P\_EXT(\cdot)$ and $N\_EXT(\cdot)$, therefore, define the generalized operators of erosion and dilation, respectively.

Theorem 2.2:

The following three conditions are equivalent:

(a) f is a positive Boolean function of n variables.

(b) $P\_EXT(on(f)) = on(f)$.

(c) $N\_EXT(off(f)) = off(f)$.

□

Note that, by Theorem 2.2, any Boolean function whose *on−set* is $P\_EXT(A)$ or whose *off−set* is $N\_EXT(A)$ is a positive Boolean function where A is a subset of *n−cube*. More properties with regard to these two operators can refer to [2].

THE CLASSIFICATION OF STACK FILTERS

Four different types of stack filters are identified in this section. They are called type-0 through type-3 and are defined in terms of characteristics of the *on−sets* of the filters in that type.

The set diagrams, shown in Figure 1, are used to clarify their characteristics. A circle is used to represent the whole space $\{0,1\}^{2N+1}$ or x^{2N+1}, the upper semicircle is used to represent the subspace $x^N 1 x^N$, and the lower semicircle is used to represent the subspace $x^N 0 x^N$. Note that the highest and the lowest points in this diagram are 1^{2N+1} and 0^{2N+1}, respectively. One is the largest vertex and the other is the smallest vertex in $\{0,1\}^{2N+1}$. The shaded areas in Figure 1 represent possible case *on−sets* of some positive Boolean functions.

Type-0 Stack Filters

If the *on−set* of f is the empty set, $x^N 1 x^N$, or x^{2N+1}, the stack filter $S_f(\cdot)$ is called a type-0 stack filter. The set diagrams of *on−set* on which type-0 stack filters are based are shown in Figure 1 (a).

The equivalent Boolean expression of f is 0, x_{N+1}, or 1.

Clearly, a stack filter of this type performs only trivial operations. They perform the identity operation if $on(f)$ is $x^N 1 x^N$; they map all input signals to 1^L after one pass if $on(f)$ is x^{2N+1}; or they map all signals to 0^L after one pass if $on(f)$ is the empty set.

Type-1 Stack Filters

If the *on−set* of f is a proper subset of $x^N 1 x^N$, then the stack filter $S_f(\cdot)$ is called a type-1 stack filter. A typical set diagram of the *on−set* on which type-1 stack filters are based is shown in Figure 1 (b). The convergence property and behavior of type-1 stack filters are discussed in [2].

The equivalent Boolean expression of f is

$$f(x_1, x_2, \ldots, x_{2N+1}) = x_{N+1} g(x_1, x_2, \ldots, x_N, x_{N+2}, \ldots, x_{2N+1})$$

where g is a nontrivial positive Boolean function of $2N$ variables.

For example, the stack filter $S_f(\cdot)$ based on $f(x_1,x_2,x_3) = x_2(x_1+x_3)$ is a type-1 stack filter. Using the set approach, the *on−set* of f is $x1x \cap (1xx \cup xx1)$. Therefore, it is a subset of $x1x$.

Type-2 Stack Filters

If the *on−set* of f is a proper superset of $x^N 1 x^N$, then the stack filter $S_f(\cdot)$ is called a type-2 stack filters. A typical set diagram of *on−set* on which type-2 stack filters are based is shown in Figure 1 (c). The convergence property and behavior of type-2 stack filters is also discussed in [2].

The equivalent Boolean expression of f is $f(x_1, x_2, \ldots, x_{2N+1}) = x_{N+1} + g(x_1, x_2, \ldots, x_N, x_{N+2}, \ldots, x_{2N+1})$ where g is a nontrivial positive Boolean function of $2N$ variables.

For example, the stack filter $S_f(\cdot)$ based on $f(x_1,x_2,x_3) = x_2+x_1x_3$ is a type-2 stack filter. Using the set approach, the *on−set* of f is $x1x \cup 1x1$. Therefore, it is a superset of $x1x$.

Type-3 Stack Filters

If the stack filters are not type-0 , type-1, or type-2, then they are classified into type-3 stack filters. A typical set diagram of *on−set* on which type-3 stack filters are based is shown in Figure 1 (d). Type-3 stack filters do not all possess the convergence property. Specific examples which exhibit oscillatory behavior are given in [2].

The equivalent Boolean expression of f is either $f(x_1, x_2, \ldots, x_{2N+1}) = x_{N+1} g(x_1, x_2, \ldots, x_N, x_{N+2}, \ldots, x_{2N+1}) + h(x_1, x_2, \ldots, x_N, x_{N+2}, \ldots, x_{2N+1})$ or $f(x_1, x_2, \ldots, x_{2N+1}) = g(x_1, x_2, \ldots, x_N, x_{N+2}, \ldots, x_{2N+1})$ where g and h are nontrivial positive Boolean functions of $2N$ variables.

Examples and Decomposition

One of the first questions that comes to mind is how the various rank-order filters would be classified according to the above four types. The positive Boolean function on which the *nth* rank-order filter is based is

$$f(x_1, x_2, \ldots, x_{2N+1}) = \sum_{1 \le i_1 < i_2 < \cdots < i_n \le 2N+1} x_{i_1} x_{i_2} \cdots x_{i_n}.$$

It is not difficult to show that the $(2N+1)th$ rank-order (the minimum) filter is a type-1 stack filter, 1st rank-order (the maximum) filter is a type-2 stack filter, and all other rank-order filters are actually type-3 stack filters.

Any type-3 stack filter can be decomposed into a type-1 stack filter and a type-2 stack filter. This decomposition is shown in the following property and is heavily used in Section 5.

Property 3.1:

If $S_f(\cdot)$ is a type-3 stack filter, then

$$f(x_1, x_2, \ldots, x_{2N+1}) = g(x_1, x_2, \ldots, x_{2N+1}) + \bar{x}_{N+1} h(x_1, x_2, \ldots, x_{2N+1}) \quad (3.1)$$

where $on(g) = (on(f) \cap x^N 1 x^N)$ and $on(h) = (on(f) \cap x^N 0 x^N) \cup x^N 1 x^N$.

□

Property 3.1 can be easily derived from the set diagram of type-3 stack filters.

ROOT-PRESERVING LEMMA

In this subsection, we state the binary version of the *root-preserving lemma.* Before we state this lemma, we need to define some notation.

Definition 4.1:

The *one−set* obtained with a window of width $2N+1$ from a specific binary signal X is the set of subsignals of length $2N+1$ which are contained in the appended signal of X and whose center point has value "1". We denote it as $one(X; 2N+1)$. The *zero−set* obtained with a window of width $2N+1$ from a specific binary signal X is the set of subsignals with length $2N+1$ which are contained in the appended signal of X and whose center point has value "0". We denote it as $zero(X; 2N+1)$.

□

These sets can be represented as follows:

$$one(X;2N+1) = \{ (x'_{i-N}, x'_{i-N+1}, \ldots, x'_i, \ldots, x'_{i+N}) : x'_i = 1,\ i \in \{1,2,\ldots,L\} \}$$

and

$$zero(X;2N+1) = \{ (x'_{i-N}, x'_{i-N+1}, \ldots, x'_i, \ldots, x'_{i+N}) : x'_i = 0,\ i \in \{1,2,\ldots,L\} \}.$$

Example 4.1:

Let $X =$ 11001001 and consider a window of width 3. The appended signal corresponding to X is 1110010011. Note that for each end there is a "1" appended.

one−set of X with window of width 3

$= one(X; 3) = \{ 111, 110, 010, 011 \}$

zero−set of X with window width 3

$= zero(X; 3) = \{ 100, 001, 100, 001 \} = \{ 100, 001 \}$

□

Now, we state the root-preserving lemma as follows:

Lemma 4.1:[2]

A binary stack filter $S_f(\cdot)$ based on the positive Boolean function f of $2N+1$ variables preserves a set of specific binary signals A if and only if $\bigcup_{X\in A} one(X; 2N+1) \subseteq on(f)$ and $\bigcup_{X\in A} zero(X; 2N+1) \subseteq off(f)$.

□

For convenience, we let $one(A; 2N+1) = \bigcup_{X\in A} one(X; 2N+1)$ and $zero(A; 2N+1) = \bigcup_{X\in A} zero(X; 2N+1)$.

ROOT-PRESERVING PROPERTY AND ASSOCIATIVE MEMORY CAPABILITY

The associative memory capability using the approach of stack filters is investigated in this section. We consider the root set of a stack filter to be an associative memory.

In Section 5.1, we present some root-preserving properties of type-1 through type-3 stack filters. In Section 5.2, we define the concept of a *better filter* to illustrate the difference between associative memories constructed from two stack filters which preserve a set of specific roots (desired patterns) and we also show some properties related to better filters.

Root-Preserving Properties under the Classification of Stack Filters

In the sequel, we assume the length of all signals is L and the window width of any stack filter that is being considered is $2N+1$.

The set of roots of the stack filter $S_f(\cdot)$, called the root set, is denoted as $R(f)$; that is,

$$R(f) = \{ X: X \in \{0, 1\}^L \text{ and } S_f(X) = X \}. \tag{5.1}$$

Now, we present some properties about the root sets of stack filters.

Property 5.1:

Let $S_f(\cdot)$ and $S_g(\cdot)$ be two type-1 stack filters. If $on(f) \subseteq on(g)$, then $R(f) \subseteq R(g)$.

□

Property 5.2:

Let $S_f(\cdot)$ and $S_g(\cdot)$ be two type-2 stack filters. If $off(f) \subseteq off(g)$, then $R(f) \subseteq R(g)$.

□

Property 5.3:

Let $S_f(\cdot)$ and $S_g(\cdot)$ be two type-3 stack filters. If $(on(f) \cap x^N 1 x^N) \subseteq (on(g) \cap x^N 1 x^N)$ and $(off(f) \cap x^N 0 x^N) \subseteq (off(g) \cap x^N 0 x^N)$, then $R(f) \subseteq R(g)$

□

Recall that any type-3 stack filter $S_f(\cdot)$ can be decomposed into a type-1 stack filter $S_g(\cdot)$ and a type-2 stack filter $S_h(\cdot)$, as shown in Property 3.1. Then, the root-preserving behavior of $S_f(\cdot)$ with regard to $S_g(\cdot)$ and $S_h(\cdot)$ is shown as follows:

Property 5.4:

$R(f) \subseteq R(g)$ and $R(f) \subseteq R(h)$.

□

Properties 5.1 through 5.4 can be easily shown by using the *root–preserving lemma.*

Better Filters

If a class of stack filters are designed to preserve a set of specific signals, there exists a stack filter whose root set is very close to this set of specific signals. By this consideration, we define the concept of *better filter* as follows:

Definition 5.1:

Suppose that stack filters $S_f(\cdot)$ and $S_g(\cdot)$ both preserve a set of specific signals, A. We say "$S_f(\cdot)$ is better than $S_g(\cdot)$ with respect to A" if $\#(R(f)) \leq \#(R(g))$ where $\#(\cdot)$ is the cardinality of a set. Note that A is a subset of $\{0, 1\}^L$.

□

Thus, the *best* stack filter with respect to a given set A has the smallest number of root signals which are *not* in set A. Those root signals are considered to be "spurious" with respect to set A; they can also be called false memories [5].

Property 5.5:

The stack filter $S_f(\cdot)$, based on the positive Boolean function f whose *on–set* is $P\_EXT(one(A; 2N{+}1))$, is the *best* filter among all the type-1 stack filters with respect to A.

Proof:

By the *root–preserving lemma,* $S_f(\cdot)$ preserves A. Hence, we also need to show that $S_f(\cdot)$ is the best filter among the type-1 stack filters which also preserve A. Let $S_g(\cdot)$ be any type-1 stack filter which preserves A.

First, we show that $on(f) \subseteq on(g)$. Since $S_g(\cdot)$ preserves A, $one(A; 2N{+}1) \subseteq on(g)$ by root-preserving lemma. By the property of positive extension operator and Theorem 2.2, $on(f) = P\_EXT(one(A; 2N{+}1)) \subseteq P\_EXT(on(g)) = on(g)$.

Finally, by Property 5.1, $R(f) \subseteq R(g)$; that is, $S_f(\cdot)$ is the best filter among the type-1 stack filters with respect to A.

□

Example 5.1:

Let set A be the monotonic signals of length 15; i.e., $L=15$, and $2N+1=5$. That is,

$$A = \{0^n 1^{15-n} : n = 0, 1, \ldots, 15\} \cup \{1^n 0^{15-n} : n = 0, 1, \ldots, 15\}.$$

$one(A;5) = \{11100, 11110, 11111, 00111, 01111\}$. Thus, $P\_EXT(one(A;5)) = 111xx \cup xx111$. Then the stack filter $S_f(\cdot)$ based on $f(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 x_3 + x_3 x_4 x_5$ is the best stack filter among the type-1 stack filters which preserve the monotonic signals.

□

Property 5.6:

The stack filter $S_f(\cdot)$, based on the positive Boolean function f whose *off*$-$*set* is $N\_EXT(zero(A; 2N+1))$, is the *best* filter among all the type-2 stack filters with respect to A.

Proof:

By the same manner used in the previous property, this property clearly holds.

□

Example 5.2:

Let A, L, and $2N+1$ be same as shown in the preceding example.

$zero(A;5) = \{11000, 10000, 00000, 00011, 00001\}$. Thus, $N\_EXT(zero(A;5)) = xx000 \cup 000xx$. Then the stack filter $S_f(\cdot)$ based on $f(x_1, x_2, x_3, x_4, x_5) = \overline{x_3}\,\overline{x_4}\,\overline{x_5} + \overline{x_1}\,\overline{x_2}\,\overline{x_3}$ is the best stack filter among the type-2 stack filters which preserve the monotonic signals. Note that after it is simplified $f(x_1, x_2, x_3, x_4, x_5) = x_3 + x_1 x_4 + x_1 x_5 + x_2 x_4 + x_2 x_5$.

□

In Properties 5.5 and 5.6, we have proposed two learning schemes to store the set of desired patterns A : $on(f) = P\_EXT(one(A; 2N+1))$ and $off(f) = N\_EXT(zero(A; 2N+1))$. Their introduction and primitive properties were shown in [2].

In the following, we will present two classes of type-3 stack filters which are better than any type-1 or type-2 stack filters with respect to A.

First, we define two operators which will be used to change a set C in $\{0, 1\}^{2N+1}$ to another set in $\{0, 1\}^{2N+1}$ as follows:

$$D^0(C) = \{u0v : \forall\ u1v \in C\}$$

and

$$D^1(C) = \{u1v : \forall\ u0v \in C\}.$$

The operator $D^0(\cdot)$ is used to mirror a vertex in the subspace $x^N 1 x^N$ to the subspace $x^N 0 x^N$ and the operator $D^1(\cdot)$ is used to mirror a vertex in the subspace $x^N 0 x^N$ to the subspace $x^N 1 x^N$.

Now, two much better classes of type-3 stack filters which can be found and shown as follows:

Property 5.7:

If $D^0(P\_EXT(one(A; 2N+1))) \cap (x^N 0 x^N - N\_EXT(zero(A; 2N+1))) \neq \emptyset$ and let set B be any nonempty subset of $D^0(P\_EXT(one(A; 2N+1))) \cap (x^N 0 x^N - N\_EXT(zero(A; 2N+1)))$, then the stack filter $S_f(\cdot)$, based on the positive Boolean function f whose *on*$-$*set* is $P\_EXT(one(A; 2N+1) \cup B)$, is better than any type-1 stack filters with respect to A.

Proof:

By the construction, $on(f) \cap x^N 1 x^N = P\_EXT(one(A; 2N+1))$ and $N\_EXT(zero(A; 2N+1)) \subseteq off(f)$. Thus, $S_f(\cdot)$ preserves A.

Since $S_f(\cdot)$ is a type-3 stack filter whose type-1 decomposition is $P\_EXT(one(A; 2N+1))$, by Property 5.4, $R(f)$ is a subset of the root set of the stack filter which we constructed in Property 5.5. Therefore, $S_f(\cdot)$ is better than any type-1 stack filters with respect to A.

□

Example 5.3:

Let A, L, and $2N+1$ be same as shown in Example 5.1. $D^0(P\_EXT(one(A;5)))=110xx \cup xx011=B_1$. $(x^2 0 x^2 - N\_EXT(zero(A;5)))=\{11001,11010,11011, 10001,10010,10011, 01001,01010,01011\} = B_2$. Then, $B_1 \cap B_2 = \{01011,10011,11001,11010,11011\}$. If we choose $B=B_1 \cap B_2$, then $P\_EXT(one(A;5) \cup B) = xx111 \cup x1x11 \cup 1xx11 \cup 11xx1 \cup 11x1x \cup 111xx$. That is, $S_f(\cdot)$ based on $f(x_1,x_2,x_3,x_4,x_5) = x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_4x_5 + x_2x_4x_5 + x_3x_{x_4}x_5$ is better than the stack filter we constructed in Example 5.1.

□

Property 5.8:

If $D^1(N\_EXT(zero(A;2N+1))) \cap (x^N 1 x^N - P\_EXT(one(A;2N+1))) \neq \emptyset$ and let set B be any nonempty subset of $D^1(N\_EXT(zero(A;2N+1))) \cap (x^N 1 x^N - P\_EXT(one(A;2N+1)))$, then the stack filter $S_f(\cdot)$, based on the positive Boolean function f whose *off−set* is $N\_EXT(zero(A;2N+1) \cup B)$, is better than any type-2 stack filters with respect to A.

Proof:

By the same manner used in the previous property, this property also holds.

□

Example 5.4:

Let A, L, and $2N+1$ be same as shown in Example 5.1. $D^1(N\_EXT(zero(A;5)))=xx100 \cup 001xx=B_1$. $(x^2 1 x^2 - P\_EXT(one(A;5)))=\{00100,00101,00110, 01100,01101,01110,10100,10101,10110\} = B_2$. Then, $B_1 \cap B_2 = \{00100,00101, 00110,01100,10100\}$. If we choose $B=B_1 \cap B_2$, then $N\_EXT(zero(A;5) \cup B) = 000xx \cup 00x0x \cup 00xx0 \cup 0xx00 \cup x0x00 \cup xx000$. That is, $S_f(\cdot)$ based on $f(x_1,x_2,x_3,x_4,x_5) = x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 + x_1x_2x_3 + x_3x_4x_5$ is better than the stack filter we constructed in Example 5.2.

□

If we choose the stack filter which is one of the two filters constructed in Properties 5.7 and 5.8 and has the smaller cardinality of root sets, then it is better than any type-1 and type-2 stack filters.

CONCLUSION

The important issue was addressed in this paper is the root-preserving property and the capability of associative memory of stack filters. We have successfully used the stacking property of set version to clarify the root-preserving lemma and then used it to analyze the root-preserving properties under the classification of stack filters. We also introduced a new concept, called *better* filter, to analyze the problem of associative memory capability -- what is the better (best) stack filter which can preserve a set of specific signals ?

REFERENCES

[1] P.D. Wendt, E.J. Coyle, and N.C. Gallagher, "Stack filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-34, no. 4, August 1986.

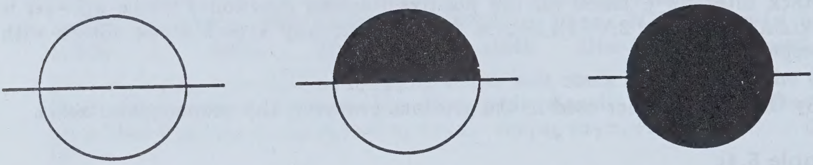
[2] P.-T. Yu and E.J. Coyle, "Convergence behavior and N-roots of stack filters," accepted by *IEEE Trans. on Acoustics, Speech, and Signal Processing.*

[3] N.C. Gallagher and G.L. Wise, "A theoretical analysis of the properties of median filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-29, no. 6, Dec. 1981.

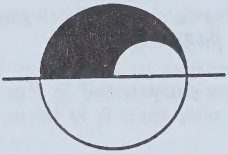
[4] S.G. Tyan, "Median filtering: deterministic properties," in *Two Dimensional Digital Signal Processing II: Transforms and Median Filters,* T.S. Huang, Ed. New York: springer-Verlag, 1981.

[5] B.L. Montgomery and B.V.K. Vijaya Kumar, "Evaluation of the use of the Hopfield neural network model as a nearest-neighbor algorithm," *Applied Optics,* vol. 25, no. 20, Oct. 1986.

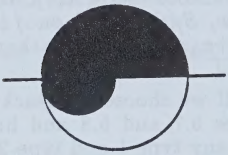
[6] P.A. Maragos and R.W. Schafer, "Morphological filters - part I: the set theoretic analysis and relations to linear shift invariant filters." and "Morphological filters - part II: their relations to median, order statistic and stack filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing,* vol. ASSP-35, no. 8, Aug. 1987.



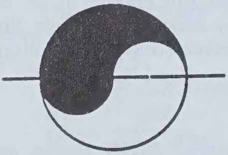
(a) type-0



(b) type-1



(c) type-2



(d) type-3

Figure 1 : The classification of stack filters.

Non-Linear Description of Signals Using Mathematical Morphology

Aldo Morales and Raj Acharya

Department of Electrical and Computer Engineering
State University of New York at Buffalo
Amherst, N.Y. 14260

Summary

Linear multiscale filtering has been applied to describe signals using primitives such as local extrema and its derivatives [14]. The local extrema represents a distinct feature that is mapped to a space since the scale of the kernel is continous. The kernel used is well-behaved in the sense that it is symmetric and strictly decreasing about the mean. Furthermore, once a feature (zero-crossing) appears, as the scale decreases, that feature will remain. However care must be taken because phantom zero-crossings may appear [5], causing a false alarm detection in which case a reduced scale map should be used.

Our approach is to develop a new non-linear description of signals using Mathematical Morphology (**M.M.**). Different methods have been developed in that the size (scale) of the structuring element is used to detect features in signals and images. Examples are size distribution of morphological openings [12], Quench function [4], propagation function [4], alternating sequential filters [13], Pattern Spectrum [8] and adaptive structuring elements [1],[9].

It is well-known that multiscale opening non-linearly smooths the images by probing it with different structuring elements at all scales while preserving the edges. This procedure is axiomatic in nature because the structuring element must be convex, compact and homothetic; using Matheron's axioms for size distribution it was proved that multiscale opening is an anti-extensive, increasing and stable operation [6], [12]. In a recent paper [3], the monotonicity criteria was studied, without directly using these three axioms.

Due to the advantages of multiscale opening in preserving the edges and the monotonicity criteria, we propose the following mapping

$$F(x) = \nabla^2(f \circ kB)(x) = 0$$

where ∇^2 is the second derivative of $f \circ kB$, which is the opening of f with the structuring element **B** at the scale k. This mapping was proposed by [10], and also appeared in a recent paper [8].

However a problem still remains: how to chose the structuring element **B**. We pose this as an optimization problem [10]. To define **B** from the space of convex, compact and homothetic structuring elements we follow an approach similar to that of [2], i.e. the **SE** should be chosen subject to three constraints : 1) Minimization of false detection of local extrema, 2) Good localization and 3) Only one response to a single edge.

Since morphological multiscale opening is a highly nonlinear mapping, in general it will be difficult to find a closed form solution for **B**. In this paper, we provide

numerical solution to the above problem. A series of experiments are carried out for an ideal step edge with different signal-to-noise ratio **SNR**, different **SEs** and different scales. Results based on these experiments will be provided.

In some recent literature [6],[7] it was proved that there is a link between morphological operations and linear filtering, making it feasible now to analyze multiscale linear filtering through **M.M.** . The natural way of doing this is by using basis functions. In this paper we will report (see [11]) that the basis functions generated by the Gaussian kernel are not convex nor homothetic but finite and well bounded.

References

[1] J. Beucher, J.M. Bolsevillle, F. Lenoir, "Traffic Spatial Measurement Using Video Image Processing", *SPIE Cambridge 87 Symp. on Advances in Intelligent Robotics Systems*, Cambridge Nov. 1987.

[2] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 8, No. 6, Nov. 1986.

[3] M. H. Chen and P. F. Yan, "A Multiscaling Approach Based on Morphological Filtering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 11, No. 7, July 1989.

[4] J.L. Chermant, M. Coster and G. Gougeon, "Shape Analysis in R^2 Space Using Mathematical Morphology", *Journal of Microscopy*, Vol. 145, Pt.-2, February 1987.

[5] J. Clark, "Singularity Theory and Phantom Edges in Scale Space", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 10, No. 5, September 1988.

[6] C. Giardina and E. Dougherty, *Morphological Methods in Image and Signal Processing*, New Jersey, Prentice Hall, 1988.

[7] P. Maragos and R. W. Schaffer, " Morphological Filters-Part I: Their Set Theoretic Analysis and Relations to Linear Shift-Invariant Filters", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-35 , August 1987.

[8] P. Maragos, "Pattern Spectrum and Multiscale Shape Representation" , *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 11, No. 7, July 1989.

[9] A. Morales, "Adaptive Structuring Element for Noise and Artifact Removal", *23d Annual Conference in Information Sciences and Systems.* The John Hopkins University. Baltimore MD. Mar 1989 pp. 211-216.

[10] A. Morales, Ph. D. Abstract, "Non-linear Description of Signals Using Mathematical Morphology", *SUNY at Buffalo*, May 1989.

[11] A. Morales and Raj Acharya, "Convexity, Compactness and Bounds of Basis Functions for Linear Filters", *Internal Report, SUNY at Buffalo*, May 1989.

[12] J. Serra, *Image Analysis and Mathematical Morphology.* New York, Academic Press, 1982.

[13] J. Serra, *Image Analysis and Mathematical Morphology, Vol II*, New York, Academic Press 1988.

[14] A. Witkin, "Scale Space Filtering", in *Proc. 8th Int. Joint Conf. Artificial Intelligence*, Karlsruhe, 1983, pp. 1019-1022.

SHAPE RECOGNITION BY MAXIMUM LIKELIHOOD METHODS†

KIE-BUM EOM AND JUHA PARK
Department of Electrical Enginering and Computer Science
The George Washington University
Washington, D.C. 20052

Abstract

In this paper, we considered the recognition of unknown shapes by maximum likelihood methods. The contour of a shape is represented by its centroidal profile, and it is fitted by a circular autoregressive model. Two different shape recognition problems are considered; the decision on the similarity of two unknown shapes, and the classification of an unknown shape as one of many known shapes. Maximum likelihood decision rules for these two cases are derived. The decision rules are invariant to translation, rotation, and size change after normalizing the estimates.

The developed algorithms are applied to classify eight classes of machine parts and eight classes of aircraft shapes. For each class, 60-80 samples are generated by rotating and dilating the original shape. In the experiment, more than 98% of machine parts are classified correctly, and more than 97% of aircraft shapes are correctly classified. This result is better than previous model based approaches.

1. Introduction

The recognition of two dimensional shapes is an important area of pattern recognition. Since most of planar shapes, such as aircraft, machine parts, characters, graphic symbols, etc, can be represented by their closed contours, many contour based algorithms have been suggested. Traditionally, moment based methods [3], transform based methods, or method based on curvature function change [2] are widely used. Recently, a model based approach is proposed [7], and a good experimental performance is reported [1].

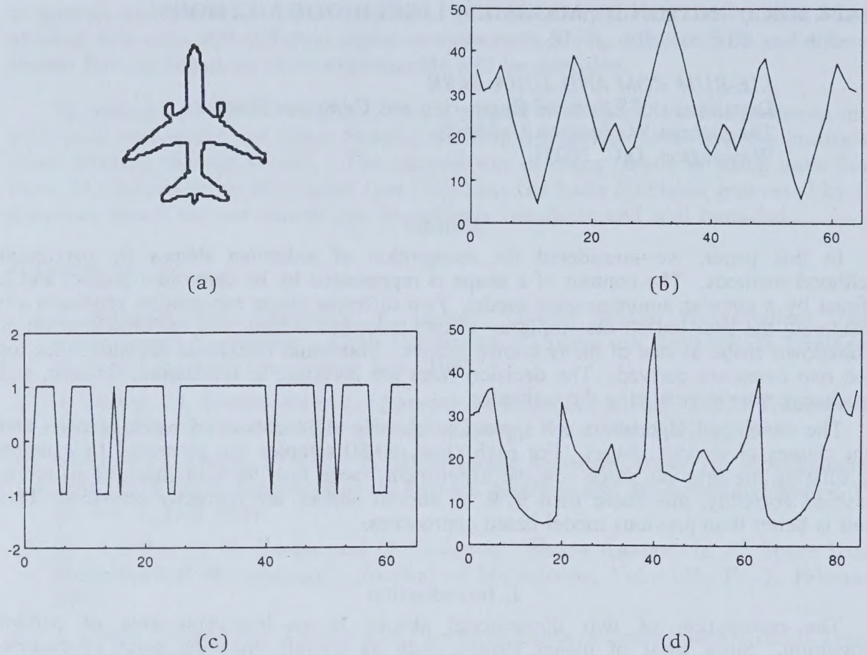
Even if the model based method suggested in [1][7] successfully classifies different machine parts, the performance on more complex shapes such as aircraft shapes is not satisfactory. It is because the use of inadequate sampling method and nonoptimal classifier. The contour is sampled by radius vectors from the centroid which are angularly equispaced is a nonuniform sampling method. Therefore, if the contour is near parallel to radius vector, then it is undersampled, and if the contour is perpendicular to radius vector, then it is oversampled. This is especially true, when the shape has long narrow parts such as nose of an aircraft. Also the classifier based on weighted distance measure used in [1] is not optimal and is not scale invariant unless the weight matrix is normalized.

To avoid nonuniform sampling problem of the angularly equispaced angular vector sampling method, we used the boundary profile as a one dimensional representation of a closed contour. The boundary profile is defined as the sequence of distances from the centroid to boundary points when the contour is followed in constant speed. This is the magnitude of centroidal profile defined by H. Freeman [2]. The boundary profile is a unique representation of contours, when it is used with the phase information. Boundary profile of a contour can be easily computed from its chain code string [2]. The figure 1 shows an example of the boundary profile representation.

Of course, there exist other methods which uniformly sample contours, such as curvature function. However, the estimation of curvature function of a digital contour is difficult and computationally expensive. Approximation methods such as polygonal approximation cannot be applied to small size contours. We used the boundary profile of a contour after resampling it to 64 data points.

---

† Partially supported by the National Science Foundation under the grant IRI-8809391.



(a) (b)

(c) (d)

Figure 1. Boundary profile representation of a contour. (a) Original closed contour, (b) Boundary profile defined by the lengths of radius vectors. (c) Phase of boundary profile which is positive if the direction of radius vector is counter clockwise, negative otherwise. (d) Data sampled by unwrap-and-stretch method in [1]. The sharp peaks are from the undersampling of the contour.

The boundary profile of a contour is a periodic time series. The characteristics of the contour can be extracted by fitting it by a circular autoregressive model. The parameters of the circular autoregressive model are invariant under rotation and translation. The dilation invariance can be achieved by normalizing them. We will derive the normalized parameter estimates, and their properties will be investigated in section 2.

We developed methods to recognize shapes by maximum likelihood decision rules after representing them by autoregressive models. Two different cases are considered. First, we developed a decision rule to test if two unknown shapes are identical. Then, we developed a decision rule to classify an unknown object as one of many known objects. The decision rules are given in terms of maximum likelihood estimates and are invariant under linear transformations, such as translation, rotation, and dilation.

For determining the similarity of two shapes, we first compute the likelihood functions of two hypotheses H_0 and H_1. Under the hypothesis H_0, two shapes are assumed the same, and under the hypothesis H_1, two shapes are assumed different. The likelihood functions $p(\mathbf{r}|H_0)$ and $p(\mathbf{r}|H_1)$ are computed in terms of estimates of model parameters. The log-likelihood ratio $\log p(\mathbf{r}|H_0) - \log p(\mathbf{r}|H_1)$ is also computed. The log-likelihood ratio can be considered as an inverse measure of distance between two shapes. Therefore, the two shapes are decided as the same shape if the log-likelihood ratio is large, and as different shapes if the log-likelihood ratio is small.

The classification of an unknown shape as one of many known shape classes by maximum likelihood method is developed. The number and characteristics of shape classes are assumed known. Each class of shape is represented by its circular autoregressive model parameters. The likelihood function $p(\mathbf{r}|C_i)$ for shape class C_i is given in terms of parameter estimates and the parameters for the shape class C_i. Then the classification is made by the maximum likelihood principle.

The contour recognition algorithm developed in this paper is applied to recognizing machine parts and aircraft shapes. 1,120 test samples are generated from eight different machine parts and eight different aircraft shapes by rotating and dilating them. The boundary profile of each contour is resampled to 64 points, and a circular autoregressive model is fitted to estimate parameters. In the experiment to classify machine parts of eight different classes, more than 98% of samples of different sizes and orientations are classified correctly. In the experiment with aircraft shapes of eight different classes, more than 97% of samples of different sizes and orientations are classified correctly. The method suggested in [1] was implemented and tested with the same data explained in the above. The result of the method in [1] is worse than the result of the method suggested in this paper, and the performance of the method presented in this paper is much better than previous methods on aircraft shapes.

2. Circular Autoregressive Model

Let $\{r(t), t=1, \cdots, N\}$ be an observation sequence which is assumed to follow the circular autoregressive (CAR) model.

$$r(t) = \alpha + \sum_{j=1}^{m} \theta_j r(t-j) + \sqrt{\beta}\,\omega(t) \tag{2.1}$$

where m is the order of model and j are positive integers. $\{\omega(t), t=1, \cdots, N\}$ is a zero mean white Gaussian process with unit variance.

Since a boundary is closed, $\{r(t)\}$ and $\{\omega(t)\}$ are periodic with period N.

$$r(t+N) = r(t), \quad \text{for all integer } t$$

$$\omega(t+N) = \omega(t), \quad \text{for all integer } t$$

Let the parameter vector $\theta = [\theta_1, \cdots, \theta_m]^T$, and the state vector $\mathbf{z}(t-1) = [r(t-1), \cdots, r(t-m)]^T$. Then the equation (2.1) can be rewritten as following.

$$r(t) = \alpha + \theta^T z(t-1) + \sqrt{\beta}\,\omega(t) \tag{2.2}$$

The parameters $[\theta, \beta]$ are invariant to object translations, to rotations of the object and variations in the starting position. The parameter α is proportional to the mean radius vector length, and the parameter β is dependent on the scale factor. The parameters can be normalized so that they are invariant to scaling. The normalized parameters $[\theta, \beta/\alpha^2]$ are scale invariant.

Define a NxN circulant matrix $\mathbf{B}(\theta)=\text{circulant}[1,0,0,\ldots,-\theta_m,-\theta_{m-1}, \cdots, -\theta_1]$. Because the sequence $\{r(t)\}$ is periodic with period N, the joint probability density of $\mathbf{r}=[r(1), \cdots, r(N)]$ is written as following.

$$p(\mathbf{r}|\alpha,\theta,\beta) = \prod_{t=1}^{N} p(y(t)|z(t-1))$$

$$= |\det B(\theta)| \left(\frac{1}{2\pi\beta}\right)^{\frac{N}{2}} \cdot \exp\left[\frac{-1}{2\beta}\sum_{t=1}^{N}(r(t)-\alpha-\theta^T z(t-1))^2\right] \tag{2.3}$$

The maximum likelihood estimators of parameters θ, α and β are obtained by maximizing (2.3). For large N, the circulant matrix $B(\theta)$ becomes an almost triangular matrix, and $\det|B(\theta)| \approx 1$. Least square estimators are close to the true maximum likelihood estimators for large N. Define the augmented state vector $\mathbf{u}(t) = [\mathbf{z}^T(t-1), 1]^T$. Let $\hat{\theta}$, $\hat{\alpha}$, and $\hat{\beta}$ be the maximum likelihood estimators of θ, α, and β, respectively. Then maximum likelihood estimators of parameters are given by the following equations.

$$\begin{bmatrix}\hat{\theta}\\ \hat{\alpha}\end{bmatrix} = \begin{bmatrix}\sum_{t=1}^{N}\mathbf{z}(t-1)\mathbf{z}^T(t-1) & \sum_{t=1}^{N}\mathbf{z}(t-1)\\ \sum_{t=1}^{N}\mathbf{z}^T(t-1) & N\end{bmatrix}^{-1}\begin{bmatrix}\sum_{t=1}^{N}\mathbf{z}(t-1)r(t)\\ \sum_{t=1}^{N}r(t)\end{bmatrix} \tag{2.4}$$

$$\hat{S} = [\sum_{t=1}^{N}\mathbf{u}(t)\mathbf{u}^T(t)]^{-1} = \begin{bmatrix}\hat{A} & \hat{C}\\ \hat{C}^T & \hat{D}\end{bmatrix} \tag{2.5}$$

$$\hat{\beta} = \frac{1}{N}\sum_{t=1}^{N}[r(t) - \hat{\alpha} - \hat{\theta}^T z(t-1)]^2 \tag{2.6}$$

The parameter estimator $\hat{\theta}$ is scale invariant. The estimates $\hat{S}$ and $\hat{\beta}$ are invariant to rotation and dilation, and can be normalized for the scale invariance. The invariance properties of estimates are summarized in the following lemma 1.

Lemma 1: Let $\hat{\theta}$, $\hat{\alpha}$, $\hat{\beta}$, and $\hat{S}$ be least squares estimators of parameters of a circular autoregressive model as defined in (2.6). The normalized least squares estimators can be defined as following.

$$\hat{\theta}_n = \hat{\theta} \tag{2.7}$$

$$\hat{\beta}_n = \hat{\beta}/\hat{\alpha}^2 \tag{2.8}$$

$$\hat{S}_n = \begin{bmatrix}\hat{\alpha}^2\hat{A} & \hat{\alpha}\hat{C}\\ \hat{\alpha}\hat{C}^T & \hat{D}\end{bmatrix} \tag{2.9}$$

Then the normalized estimators in (2.7)-(2.9) are invariant to translation, rotation, and dilation. If a contour $\mathbf{r}_2$ is the dilated contour of $\mathbf{r}_1$, then the ratio of their parameter estimates $\frac{\hat{\alpha}_2}{\hat{\alpha}_1}$ is the ratio of dilation.

Proof: Obviously, the normalized estimates given in (2.7)-(2.9) are translation invariant, because the observation sequence $\{r(t)\}$ does not change with translation. Since the sequence $\{r(t)\}$ is circular symmetric, the estimators given in (2.4)-(2.6) does not change with rotation. Therefore, the normalized estimators given in (2.7)-(2.9) are rotation invariant.

Now, we need to show the property on dilation. Suppose that a contour $\mathbf{r}_2$ is a dilated contour of $\mathbf{r}_1$, where the dilation ratio is assumed k.

$$r_2(t) = kr_1(t), \quad t=1, \cdots, N. \tag{2.10}$$

The numbers of data points in contour $\mathbf{r}_1$ and $\mathbf{r}_1$ are the same, because the contours are sampled by the same number of sampling vectors. From (2.10),

$$z_2(t) = kz_1(t), \quad t=1, \cdots, N. \tag{2.11}$$

By the definition in (2.5), we can rewrite the block matrix $\hat{S}$ as following.

$$\hat{S} = \begin{bmatrix}\hat{A} & \hat{C}\\ \hat{C}^T & \hat{D}\end{bmatrix} = \begin{bmatrix}\sum_{t=1}^{N}z(t-1)z^T(t-1) & \sum_{t=1}^{N}z(t-1)\\ \sum_{t=1}^{N}z^T(t-1) & N\end{bmatrix}^{-1} \tag{2.12}$$

Then the relation (2.13) can be obtained after some algebraic manipulations of the result of substituting the relation (2.11) into the block matrix inversion formula (A.22, p.656 Kailath [4]).

$$\hat{S}_2 = \begin{bmatrix} \hat{A}_2 & \hat{C}_2 \\ \hat{C}_2^T & \hat{D}_2 \end{bmatrix} = \begin{bmatrix} \frac{\hat{A}_1}{k^2} & \frac{\hat{C}_1}{k} \\ \frac{\hat{C}_1^T}{k} & \hat{D}_1 \end{bmatrix} \tag{2.13}$$

Also by the definition of state vector,

$$\sum_{t=1}^{N} u_2(t) r_2(t) = \begin{bmatrix} \sum_{t=1}^{N} z_2(t-1) r_2(t) \\ \sum_{t=1}^{N} r_2(t) \end{bmatrix} = \begin{bmatrix} k^2 \sum_{t=1}^{N} z_1(t-1) r_1(t) \\ k \sum_{t=1}^{N} r_2(t) \end{bmatrix} = \sum_{t=1}^{N} u_1(t) r_1(t) \tag{2.14}$$

By substituting (2.13) and (2.14) into (2.4), we have

$$\hat{\theta}_2 = \hat{\theta}_1 \tag{2.15}$$

and

$$\hat{\alpha}_2 = k\hat{\alpha}_1 \tag{2.16}$$

By substituting (2.10),(2.11),(2.15) and (2.16) into (2.6), we have the following relation.

$$\hat{\beta}_2 \hat{\alpha}_1^2 = \hat{\beta}_1 \hat{\alpha}_2^2 \tag{2.17}$$

This finishes the proof.

□

The parameter estimates $\hat{\theta}$, $\hat{\alpha}$, $\hat{\beta}$, and $\hat{S}$ characterize the observation r(t). Also, the normalized parameters defined in lemma 1 are invariant to rotation, shift, and size change. Therefore, any decision rule involves only with these normalized parameters will also be invariant to rotation, shift, and size change. In the following sections, we will assume that all parameter estimates are normalized by the rule given in lemma 1.

3. Similarity of Two Contours

In the section 2, we have shown that a contour can be represented by a circular autoregressive (CAR) model. The parameters of the CAR model can be estimated by a least squares method, and least squares estimators are asymptotically close to maximum-likelihood estimators. The least squares estimators of CAR model parameters are invariant to rotation and translation, and can be normalized so that they are invariant to dilation.

In a typical shape recognition problem, an unknown contour needs to be classified as one of many known shapes. In the previous approaches [1][7], it is attempted to classify shapes in the parameter space by using statistical pattern classification methods, such as a minimum distance classifier. Even though these approaches perform well with perfect contours, the error rate become unacceptable with noisy contours. It is because the significance of each parameter is different for the classification. In this paper, we formulate the shape recognition problem as a maximum likelihood decision problem. In this section, we will derive a maximum likelihood decision rule to determine if two contours are identical.

Consider two contours $\mathbf{r}_1$ and $\mathbf{r}_2$.

$$\mathbf{r}_1 = [r_1(1), \cdots, r_1(N_1)]^T$$
$$\mathbf{r}_2 = [r_2(1), \cdots, r_2(N_2)]^T$$

where N_1 and N_2 are sizes of observation sequences. For the given two contours $\mathbf{r}_1$ and $\mathbf{r}_2$, the contour recognition problem is to determine whether two contours are identical or different under transformations of rotation, dilation, or translation. The following definition defines the identity of two contours under above transformations.

Definition 1 (Identity of Shapes): Two contours $\mathbf{r}_1$ and $\mathbf{r}_2$ are identical under transformation of rotation, translation and dilation if $\mathbf{r}_2$ can be obtained from $\mathbf{r}_1$ by applying combination of rotation, translation, or dilation operation. i.e.,

$$r_2(t) = ar_1(t+b), \quad a>0,\ |b|\leq N-1, \text{ for all integer } t,$$

where b is an integer.

The similarity of two contours following circular autoregressive model can be determined by their parameters. The following lemma 2 is proved by Kashyap and Chellappa [7].

Lemma 2: Suppose that two contours $\mathbf{r}_1$ and $\mathbf{r}_2$ follow circular autoregressive models with parameters $\{\alpha_1, \theta_1, \beta_1\}$ and $\{\alpha_2, \theta_2, \beta_2\}$, respectively. These two contours are identical under transformations of translation, dilation, and rotation, if and only if,

$$\theta_2 = \theta_1,$$
$$\alpha_2 = \gamma\alpha_1,$$
$$\beta_2 = \gamma^2\beta_1,$$

where γ is a constant indicating the ratio of size difference between two contours.

□

Corollary: Two contours $\mathbf{r}_1$ and $\mathbf{r}_2$ are identical if their normalized parameters $\{\theta_1, \beta_1/\alpha_1^2\}$ and $\{\theta_2, \beta_2/\alpha_2^2\}$ are the same.

□

The identity of two shapes can be tested by a statistical hypothesis testing method. Define the null hypothesis H_0 and the alternative hypothesis H_1 by the following.

H_0 : $\mathbf{r}_1$ and $\mathbf{r}_2$ are identical shapes under the definition 1.
H_1 : $\mathbf{r}_1$ and $\mathbf{r}_2$ are different shapes under the definition 1.

Then the maximum likelihood decision rule has the following form.

$$\begin{cases} \text{accept } H_0, & \text{if } \dfrac{p(\mathbf{r}_1,\mathbf{r}_2|H_0)}{p(\mathbf{r}_1,\mathbf{r}_2|H_1)} \geq K_1 \\ \text{accept } H_1, & \text{otherwise} \end{cases} \tag{3.1}$$

where K_1 is a constant. To use the above likelihood ratio test, we need to find the likelihood functions $p(\mathbf{r}_1,\mathbf{r}_2|H_0)$ and $p(\mathbf{r}_1,\mathbf{r}_2|H_1)$ to make decision by the rule given in (3.1).

First assume the hypothesis H_1. Then $\mathbf{r}_1$ and $\mathbf{r}_2$ are different shapes, and assumed to follow two different autoregressive models given in the following equations (3.2) and (3.3), respectively.

$$r_1(t) = \alpha_1 + \theta_1^T z_1(t-1) + \sqrt{\beta_1}\omega_1(t), \quad t=1,\cdots,N_1 \tag{3.2}$$
$$r_2(t) = \alpha_2 + \theta_2^T z_2(t-1) + \sqrt{\beta_2}\omega_2(t), \quad t=1,\cdots,N_2 \tag{3.3}$$

where $\omega_1(t)$ and $\omega_2(t)$ are independent zero mean white Gaussian with unit variance.

Since two contours $\mathbf{r}_1$ and $\mathbf{r}_2$ are independent, we can write as follows.

$$p(\mathbf{r}_1,\mathbf{r}_2|H_1) = p(\mathbf{r}_1|H_1)p(\mathbf{r}_2|H_1) \tag{3.4}$$

Now consider the terms $p(\mathbf{r}_i|H_1), i=1,2$. They can be obtained by solving the following integral.

$$p(\mathbf{r}_i|H_1) = \int p(\mathbf{r}_i|\phi_i,H_1)p(\phi_i,H_1)|d\phi_i| \tag{3.5}$$

where $\phi_i = [\alpha_i,\theta_i^T,\beta_i]^T$ is a vector of model parameters. The density function $p(\mathbf{r}_i|\phi_i)$ is given in (2.3), and the integral in (3.5) can be computed by the asymptotic integration.

Kashyap [6] has shown that the likelihood function $p(\mathbf{r}_i|H_1)$ is given in terms of the maximum likelihood parameter estimates. Detailed proof can be found in [6].

Therefore by the result in [6], the log-likelihood function for hypothesis H_1 is given by the following.

$$
\begin{aligned}
\log p(\mathbf{r}_1,\mathbf{r}_2|H_1) &= \sum_{i=1,2} \log p(\mathbf{r}_i|H_1) \\
&= \sum_{i=1,2} \{\log p(\hat{\phi}_i) + \frac{(-N_i+m+2)}{2}\log\hat{\beta}_i + \frac{1}{2}\log[\det(\hat{S}_i)] \\
&\quad + \frac{(-N_i+m+2)}{2}\log(2\pi) + \frac{N_i}{2} - \log N_i + \frac{1}{2}\log 2\}
\end{aligned}
\tag{3.6}
$$

where N_i is the length of sequence $\{r_i(t)\}$, $\hat{\beta}_i$ and $\hat{S}_i$ are normalized parameter estimates obtained from $\mathbf{r}_i$.

Now assume the hypothesis H_0. obtained from $\mathbf{r}_i$. Then two contours $\mathbf{r}_1$ and $\mathbf{r}_2$ are the identical shapes, and assumed to follow the models given in (3.7) and (3.8).

$$r_1(t) = \alpha_0 + \theta_0^T z_1(t-1) + \sqrt{\beta_0}\,\omega_1(t), \quad t=1, \cdots, N_1 \tag{3.7}$$

$$r_2(t) = \gamma\alpha_0 + \theta_0^T z_2(t-1) + \gamma\sqrt{\beta_0}\,\omega_2(t), \quad t=1, \cdots, N_2 \tag{3.8}$$

Since two contours $\mathbf{r}_1$ and $\mathbf{r}_2$ are identical shapes under hypothesis H_0, the parameters in (3.7) and (3.8) are the same except the scaling parameter gamma.

The likelihood function $p(\mathbf{r}_1,\mathbf{r}_2|H_0)$ can be obtained in a similar way to obtain the likelihood function for the hypothesis H_1.

$$
\begin{aligned}
\log p(\mathbf{r}_1,\mathbf{r}_2|H_0) &= \log p(\hat{\phi}_0) + \frac{(-N_1-N_2+m+2)}{2}\log\hat{\beta}_0 + \frac{1}{2}\log[\det(\hat{S}_0)] \\
&\quad + \frac{(-N_1-N_2+m+2)}{2}\log(2\pi) + \frac{N_1+N_2}{2} - \log(N_1+N_2) + \frac{1}{2}\log 2
\end{aligned}
\tag{3.9}
$$

where N_1 and N_2 are the lengths of sequences $\mathbf{r}_1$ and $\mathbf{r}_2$. The normalized estimates $\hat{\beta}_0$ and $\hat{S}_0$ are obtained by fitting the model to $\{\mathbf{r}_1, \mathbf{r}_2\}$.

Define

$$d = \log p(\mathbf{r}_1,\mathbf{r}_2|H_0) - \log p(\mathbf{r}_1,\mathbf{r}_2|H_1). \tag{3.10}$$

Then the decision rule in (3.1) can be rewritten as follows.

$$
\begin{cases}
\text{accept } H_0, & \text{if } d \geq K \\
\text{accept } H_1, & \text{otherwise}
\end{cases}
\tag{3.11}
$$

where K is a constant.

4. Classification of Shapes

Consider the classification of an unknown shape $\mathbf{r}$ as one of many known shapes. Suppose that we have M known classes, $C_1, C_2, \cdots, C_M$, of different shapes. Two shapes belong to the same class if one can be obtained by applying a linear transformation of translation, rotation, and size change from the other. If we assume that the boundary profile sequence follows a circular autoregressive model, each class C_i is represented by the normalized parameter vector $[\theta_n^i, \beta_n^i]$, where $\boldsymbol{\theta}_n^i = \boldsymbol{\theta}^i$ and $\beta_n^i = \beta^i/(\alpha^i)^2$.

Consider the likelihood function $p(\mathbf{r}|C_i)$ of the class C_i. The boundary profile $\mathbf{r}$ of an unknown contour is in the class C_i if the likelihood function $p(\mathbf{r}|C_i)$ is $p(\mathbf{r}|\theta_n^i,\beta_n^i)$, where θ_n^i and β_n^i are normalized parameters such that $\hat{\alpha}=1$. Then

$$
\begin{aligned}
p(\mathbf{r}|C_i) &= p(\mathbf{r}|\theta_n^i,\beta_n^i) \\
&= \left(\frac{1}{2\pi\beta_n^i}\right)^{\frac{N}{2}} \cdot \exp\left[\frac{-1}{2\beta_n^i}\sum_{t=1}^{N}(r_n(t)-1-\boldsymbol{\theta}_n^{iT} z_n(t-1))^2\right],
\end{aligned}
\tag{4.1}
$$

where $r_n(t) = r(t)/\alpha^i$.

The above equation (4.1) can be simplified by the method similar to the method in [5].

$$p(\mathbf{r}|C_i) = \left(\frac{1}{2\pi\beta_n^i}\right)^{\frac{N}{2}} \cdot \exp\left[\frac{-1}{2\beta_n^i}(N\hat{\beta}_n + (\hat{\theta}_n - \theta_n^i)^T \hat{S}_n^{-1} (\hat{\theta}_n - \theta_n^i)\right], \tag{4.2}$$

where normalized parameter estimates $\hat{\beta}_n$, $\hat{\theta}_n$, and $\hat{S}_n$ are obtained by normalizing the maximum likelihood estimates $\hat{\beta}$, $\hat{\theta}$, and $\hat{S}$ by the equations in (2.7)-(2.9). The maximum likelihood estimates are obtained by equations (2.4)-(2.6) from the sequence **r**.

The classification can be done by the maximum likelihood decision rule. Let $p(\mathbf{r}|C_1), p(\mathbf{r}|C_2), \cdots, p(\mathbf{r}|C_M)$ are likelihood functions for classes $C_1, \cdots, C_M$, respectively. The maximum likelihood decision rule is given by the following.

Decision rule: The unknown shape **r** belongs to the class C_i
if $p(\mathbf{r}|C_i) = \text{maximum } \{p(\mathbf{r}|C_j), j=1, \cdots, M\}$.

5. Experimental Results

To test the performance of shape recognition algorithms developed in this paper, eight different machine parts and eight different airplane shapes are selected. Figure 2 shows the contours of eight different machine parts used in this experiment. Each machine parts image is digitized in resolution of 256x256. Figure 3 shows the contours of eight different aircraft shapes used in this experiment. The aircraft shapes used are B1, Challenger, DC10, F-14, F-16, Harrier, Mig-29 and Mirage. These aircraft shapes are more complex than machine parts. Each aircraft image is digitized in resolution of 640x350.

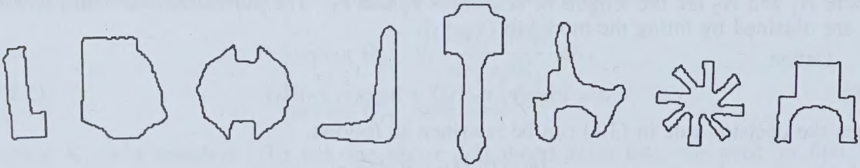


Figure 2 Contours of eight different machine parts used in the experiment.

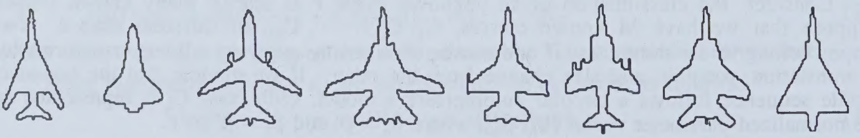


Figure 3 Contours of eight different aircraft shapes used in the experiment. From left to right; B1, Challenger, DC10, F-14, F-16, Harrier, Mig-29 and Mirage.

Test samples are made from eight machine parts shapes and eight aircraft shapes by rotating and changing the size. To generate test samples for machine parts, each machine parts image is rotated in 20 different orientations, and the size is changed to 1.0, 0.8, and 0.6 times of the original size. Therefore, 60 samples are generated for each machine part, and total of 480 samples are generated. To generate test samples for aircraft shapes, each aircraft image is rotated in 20 different orientations, and the size is changed to 1.0, 0.8, 0.6, and 0.5 times of the original size. Therefore, 80 samples are generated for each aircraft shape, and total of 640 test samples are generated.

The boundary profile of each contour is computed, and resampled to 64 points. The boundary profile is defined as the sequence of the length of radius vector from the centroid to the boundary point when the boundary is followed in a constant speed. A circular autoregressive model is fitted to resampled data, and parameter estimates are computed and normalized so that they are invariant to translation, rotation, and size change.

The similarity of two unknown shapes are tested by the decision rule developed in section 3. From the test patterns generated by rotation and size change, 200 pairs are chosen, where 100 are identical shapes under the linear transformation and 100 are different shapes. The decision rule (3.11) is applied to these 200 test pairs, and the statistics are computed. In this experiment, more than 95% of decisions were correct.

The maximum likelihood classification rule developed in section 4 is applied to both machine parts and aircraft shapes. The performance of the algorithm given in this paper is compared with the algorithm suggested by Dubois and Glanz [1]. The classification performance on machine parts is summarized in table 1.

Table 1 The classification accuarcy (in percent) of the maximum likelihood (ML) algorithm proposed in this paper and the algorithm suggested by Dubois and Glanz (DG).

class	ML	DG
1	100%	93.33%
2	100%	100%
3	100%	100%
4	95.0%	45.0%
5	93.33%	98.33%
6	100%	91.67%
7	100%	100%
8	100%	100%
Average	98.54%	91.04%

The accuracy of the maximum likelihood method on machine parts is 98.54% and the accuracy of the Dubois and Glanz's method is 91.04%. This table shows higher accuracy of the maximum likelihood approach suggested in this paper than other model based methods.

The two classification algorithms discussed in the above are applied to classify aircraft shapes. The following table 2 summarizes the result.

Table 2 The classification accuarcy (in percent) of the maximum likelihood (ML) algorithm proposed in this paper and the algorithm suggested by Dubois and Glanz (DG).

class	ML	DG
B-1	100%	63.75%
Challenger	100%	100%
DC-10	100%	60.0%
F-14	97.5%	86.25%
F-16	100%	100%
Harrier	100%	98.75%
Mig-29	82.5%	72.5%
Mirage	100%	88.75%
Average	97.5%	83.75%

The accuracy of the maximum likelihood method on aircraft shapes is 97.5% and the accuracy of the Dubois and Glanz's method is 83.75%. The difference in accuacy is more than 14%. This table shows high accuracy of the maximum likelihood approach suggested in this paper.

6. Discussions and Conclusions

We considered the contour recognition by using maximum likelihood methods with circular autoregressive models. The contour is converted to a one dimensional circular time series by computing its boundary profile. The boundary profile with phase information is a unique representation of a contour. The boundary profile of a contour is simpler to compute, yet better in performance than the sampling method by angularly equispaced radius vectors because the sampling is uniform along the contour.

Maximum likelihood decision rules to discriminate contours for two different cases are developed. The maximum likelihood decision rules are given in terms of normalized parameter estimates, and are invariant to shift, rotation, and size change of contours. The developed algorithms were tested with large number of samples of machine parts and aircraft shapes of different size and orientation. In this experiment, the performance of the algorithm developed in this paper is much better than other model based methods.

References

[1] S.R. Dubois and F.H. Glanz, ''An Autoregressive Model Approach to Two-Dimensional Shape Classification,'' *IEEE Trans. on Pattern Anal. and Machine Intell.,* Vol. PAMI-8, No. 1, Jan. 1986.

[2] H. Freeman, ''Shape Description via the Use of Critical Points,'' *Proceedings of Pattern Recognition of Image Processing,''* pp. 168-174, 1977.

[3] M.K. Hu, '' Visual Pattern Recognition by Moment Invariants,'' *IEEE Trans. on Information Theory,* Vol. IT-8, pp.179-187, Feb. 1962.

[4] T. Kailath, *Linear Systems,* Prentice Hall, 1980.

[5] R.L. Kashyap, ''Optimal Feature Selection and Decision Rules in Classification Problems with Time Series,'' *IEEE Trans. on Information Theory,* Vol. IT-24, No. 3, pp.281-288, May 1978.

[6] R.L. Kashyap, ''Optimal Choice of AR and MA Parts in Autoregressive Moving Average Models," *IEEE Trans. on Pattern Anal. and Machine Intell.,* Vol. PAMI-4, pp.99-104, March 1982.

[7] R.L. Kashyap and R. Chellappa, ''Stochastic Models for Closed Boundary Analysis: Representation and Reconstruction,'' *IEEE Trans. on Information Theory,* Vol. IT-27, No. 5, Sept. 1981.

MOTION FIELD SEGMENTATION FOR IMAGE SEQUENCE CODING

MICHAEL ORCHARD †
Department of Electrical Engineering
Princeton University, Princeton, NJ 08544

SUMMARY

Most methods for coding image sequences at low to medium bit rates are based on some form of block based motion compensation (e.g. [1]). In this approach to coding, the transmitter computes the motion of blocks of pixels from the previous frame into the current frame and sends these motion vectors to the receiver. The receiver uses the motion vectors to construct an estimate of the current frame from the previously decoded frame. The transmitter then sends additional information (compensated frame difference) to correct errors in the receiver's estimate. This paper addresses the problem of artifacts generated by block based motion compensation methods along borders of moving objects. Blocks falling on these borders contain objects moving with different velocities and compensating these blocks with a single motion vector produces a poor estimate of the current frame. A large percentage of the available bit rate is usually devoted to correcting these bad blocks, thereby reducing the overall image quality. Puri et al. [2] have proposed using smaller block sizes on borders of moving objects to reduce these artifacts. This approach requires that additional motion vectors be sent to the receiver to accommodate the increased number of blocks and the authors report no significant increase in overall coding efficiency. Others have proposed identifying objects in the image and sending object motion vectors to the receiver in place of block vectors. These approaches require large amounts of computation and often cannot deal with scenes which are too complex.

This paper proposes a method for reducing block artifacts at edges of moving objects by computing a local motion field segmentation at each block. In order to minimize the amount of side information required to be sent to the receiver and to reduce the computational complexity, the method makes the following assumptions about the image scenes being segmented. First, the algorithm is assumed to operate on high SNR images (i.e. large object intensity differences relative to digitization noise). Blocks containing multiple objects with small intensity differences do not generate very serious artifacts. Secondly, all pixel velocities in a block are assumed to match at most two different motion vectors. One of these is the vector assigned to the block and the other is selected from the vectors assigned to the 4 nearest neighboring blocks. Finally, smoothness constraints are applied to the object shapes found within blocks, reflecting the assumption that the block size is small compared to the size of objects in the scene. The block segmentation is modeled as a Markov random field with the additional constraints: (a) objects within blocks are connected and (b) objects moving with a neighboring block's motion vector are connected to an object in that block.

Based on these assumptions, an algorithm is defined for segmenting the motion field of a block. First, the motion vector (MV), from those assigned to 4 neighboring blocks, which produces the largest reduction in compensated frame difference when applied to pixels within the block is denoted the alternate MV. Second, each pixel in the block is assigned costs equal to the frame difference resulting from compensating with the block MV and the alternate MV. MLE and MAP estimates of the segmentation are considered. The MLE assigns each pixel to the MV with minimizes the cost. The MAP applies smoothness constraints to assign costs to object shapes and minimizes the sum of the total pixel costs and object costs. The receiver then compensates each pixel with the MV assigned to it by the segmentation.

---

† This work was supported by an AT&T Graduate Fellowship and was partially performed at AT&T Bell Laboratories, Holmdel NJ

Two types of coding systems using motion field segmentation are considered. In the feed-forward system, the transmitter computes the segmentation and sends it to the receiver. Because of the structure of the MAP segmentation estimate, it can be specified with a single boundary and efficiently encoded with a chain code. The feed-forward system minimizes receiver complexity. In the feed-back system, both transmitter and receiver compute a previous segmentation from the two previously decoded frames. At borders of moving objects, the transmitter computes motion vectors for compensating the previous segmentation to give the correct current segmentation. The receiver uses these motion vectors to construct the current segmentation and uses it to compensate the image. The feed-back system sends no additional side information or motion vectors to describe the segmentation.

Tests of the proposed method applied to the "table-tennis" sequence show significant (45%) reductions in compensated prediction error. In addition, the compensated image produced by the proposed method lacks many of the artifacts produced by the standard method. Figure 1 below shows the motion compensated frame produced by the standard method and the new method applied to a frame from "table-tennis". The results suggest that, besides reducing the compensated frame difference, the proposed method should give more visually acceptable results when limited bit-rates force heavy quantization of the compensated prediction error.

REFERENCES

[1] CCITT SGXV WP XV/4 Specialists Group on Coding for Visual Telephony Doc. #525, "Description of Ref. Model 8," June 9, 1989.

[2] A. Puri, H.M. Hang and D.L. Schilling, "Interframe Coding with Variable Block-size Motion-compensation," *Proc. IEEE Global Commun. Conf.*, Tokyo, Japan, Nov. 1987, pp. 2.7.1-2.7.5.

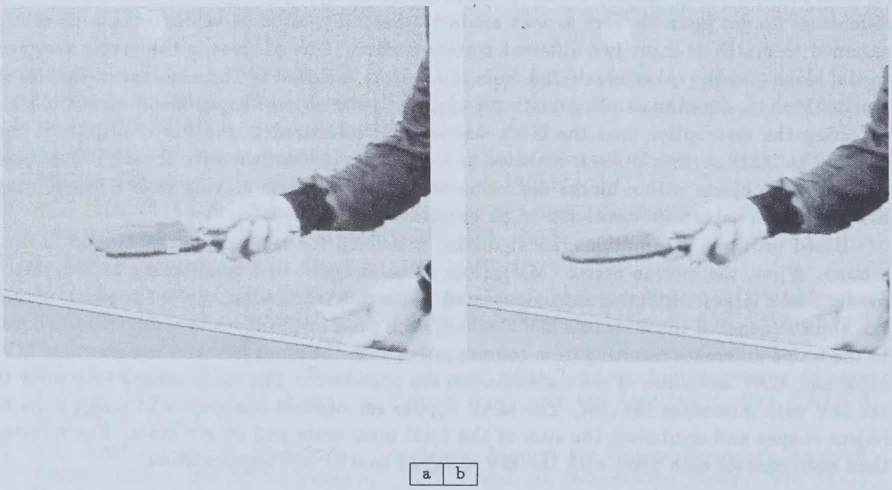


Figure 1: a) Standard compensated frame b) New compensated frame using segmentation.

MOTION COMPENSATION CODING AUGMENTED BY INTRAFRAME PREDICTION[†]

DONALD F. WILKINS
Dept. 311/0643163
McDonnell Douglas Corp.
St. Louis, MO 63166

NORMAN R. COX
Dept. of Electrical Engineering
University of Missouri-Rolla
Rolla, MO 65401

ABSTRACT

Interframe coders which make use of motion compensation techniques, often encounter difficulties in the presence of rapid motion, a scene change, or the uncovering of hidden background detail. In this paper, we show that significant improvements can result, if an intraframe prediction option is added.

INTRODUCTION

A fairly recent innovation in the area of interframe coding has been the introduction of so-called motion compensation techniques. These techniques, essentially, modify the image reference frame (i.e., the prediction signal) to account for moderate amounts of motion within the image. One approach proposed by Jain and Jain [1], simplifies computations by approximating moving areas with small rectangular blocks. This method works exceptionally well, but it is still unable to cope with (1) an abrupt scene change, (2) the presence of rapid motion, or (3) the uncovering of unknown background information. Admittedly, problems (1) and (2) seldom occur in teleconferencing and surveillance applications, and problem (3) can be resolved by storing background detail [2]. For a live television broadcast, however, a coder must accommodate all three situations. The most logical solution is to combine interframe motion-compensated coding with intraframe differential pulse code modulation (DPCM).

MODIFICATION OF THE JAIN METHOD

The Jain method basically involves dividing each incoming image frame into a fixed number of N x N blocks. For each block, a logarithmic search procedure is used to find a matching block within a corresponding (N + 2q) x (N + 2q) search area of the previous frame. For each possible displacement, a mean distortion factor is computed as follows:

$$D(i,j) = \frac{1}{N^2} \sum_{m=1}^{N} \sum_{n=1}^{N} g[U(m,n) - U_R(m+i,n+j)], \qquad -q \leq i,j \leq q$$

where $g(x)$ is a given distortion function ($g(x) = x^2$, in this case), U is the incoming block, and U_R is the corresponding block of a reference (or previous) frame. The values of i and j that yield the lowest value of D(i,j) indicate the direction of minimum distortion (DMD). At this point, the motion vector defined by i and j is usually transmitted so that the motion-compensated block can be reconstructed at the decoder. However, if the value of D(i,j) corresponding to the DMD is exceedingly large, then, it is likely that the block has experienced rapid motion, a scene change, or the uncovering of hidden background detail. By comparing $D(i,j)_{min}$ with a predetermined threshold TH, we can determine whether or not it is best to select intraframe prediction for the current block. This decision can be relayed to the receiver and decoder via one of the unused motion vector codes.

[†] This work was performed by Mr. Don F. Wilkins as a partial requirement for the Master of Science Degree, University of Missouri-Rolla.

SIMULATION RESULTS

To test this idea, we simulated the proposed system using four image sequences. Each frame of this database contained 256 x 256 picture elements (pels) with 8 bits/pel representation. Two of the sequences (CRONKITE and CHEMICAL PLANT), were identical to those used in [1]. The remaining two (CAR and KELVIN), were obtained with a computer-controlled "frame grabber." In selecting a predictor for the intraframe mode, we considered only simple algorithms that could easily be implemented at television rates. The four most-promising algorithms were then evaluated on the basis of signal error variance (σ_e^2), signal-to-noise ratio (SNR), and entropy. The best one of these was selected for the entire simulation.

As seen in the representative data given below, little improvement, if any resulted when only moderate amounts of motion were present. However, when rapid motion (panning, in this case) was simulated (PLANT 13 predicted by PLANT 9), the increase in SNR was significant. Similarly, when a scene change was simulated (PLANT 9 predicted by WALTER 5), a 3.76 dB gain in performance was achieved. Generally, it was found that the SNR and σ_e^2 almost always improved with the modified algorithm. Entropy, on the other hand, only dropped significantly (0.8 to 1.0 bits/pel) during rapid motion or a scene change. The implication of this result is that under "normal" conditions, the modified algorithm might only cause a slight decrease in the spread of the prediction error (and, hence, a decrease in quantizing noise). However, when the coder experiences rapid motion or a scene change, the potential for limiting the data rate over short periods might be useful. This strategy could, for example, serve as an alternative to subsampling and coarse quantization as a means of preventing buffer overflow during such transitions. This subject and other details are discussed in reference [3].

CASE	DATA	SNR IN DECIBELS		
		WITHOUT MOTION COMPENSATION	WITH MOTION COMPENSATION*	MODIFIED ALGORITHM*
LIMITED MOTION (UNCOVERED BACKGROUND)	WALTER 5-6	25.25	32.72	32.72
	PLANT 9-10	16.77	24.39	25.06
	CAR 3-4	30.74	31.36	31.36
	KELVIN 1-2	30.55	30.71	31.36
RAPID MOTION	PLANT 9-13	14.44	15.49	21.09
SCENE CHANGE	WALTER 5 - PLANT 9	12.10	12.25	16.01

* 32 x 32 block size.

REFERENCES

1. J. R. Jain and A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," IEEE Trans. Commun., vol. COM-29, pp. 1799-1808, December 1981.

2. N. Mukawa and H. Kuroda, "Uncovered Background Prediction in Interframe Coding," IEEE Trans. Commun., vol. COM-33, pp. 1227-1231, November 1985.

3. D. F. Wilkins, "Motion Compensation Coding Augmented by Intraframe Prediction," M.S. Thesis, University of Missouri-Rolla, May 1990.

MULTIPLE-FRAME FORMULATION OF THE PEL-RECURSIVE MOTION ESTIMATION ALGORITHM [1]

Serafim N. Efstratiadis and Aggelos K. Katsaggelos

Northwestern University
Department of Electrical Engineering and Computer Science
The Technological Institute
Evanston, Illinois 60208-3118

Abstract

In this paper, a multiple frame formulation of the pel-recursive Wiener-based motion estimation algorithm [1] and a new model-based approach to motion estimation are presented. The derivation of the algorithms is based on the assumption that both the so-called update of the initial estimate of the displacement vector and the linearization error are samples of stochastic processes. A linear least-squares estimate of the update of the initial estimate of the displacement vector from the previous frame to the current is provided, based on w observations in a causal window $\mathcal{W}$ of each of the ν previous frames. The sensitivity of the pel-recursive algorithms in the areas where occlusion occurs is studied and their performance is improved with adaptive regularization of the inverse problem that is involved. Based on our experiments with typical video-conferencing scenes, we concluded that the multiple frame Wiener-based algorithm performs better than the two-frame Wiener-based pel-recursive algorithm with respect to robustness, stability, and smoothness of the velocity field.

1 Introduction

In the field of image sequence processing much of the attention is focused on the estimation of the motion field between successive frames. The applications of the estimation of motion are numerous and vary from object tracking to spatio-temporal motion compensated image sequence filtering [6]. In the area of image sequence coding, in particular, with applications like digital TV, video-conferencing and video-phone signals, bandwidth reduction can be achieved by utilizing the interframe redundancy. In achieving this task, attention was initially focused on conditional replenishment [7]. However, an important further bandwidth reduction can be obtained by incorporating motion estimation [8]. A motion estimate, indicating the motion trajectory, can be used to compensate for movement, thereby reducing the prediction error.

A class of motion estimation techniques is represented by pel-recursive algorithms [2]. In contrast to the block-matching methods [2], pel-recursive methods allow the motion vectors to be determined at the receiver as well as at the transmitter, hence no motion vectors need to be transmitted. Another advantage is that pel-recursive methods can be much faster than the block-matching methods with the additional advantage of providing directly, that is, without the need of spatial interpolation, motion vectors with sub-pixel accuracy.

In this paper, a multiple-frame pel-recursive Wiener-based displacement estimation algorithm is presented. The algorithm is applied along the scan direction in a frame. So far all the algorithms presented in the literature for displacement estimation are based on the use of two consecutive frames. The velocity field vector at the working pixel is obtained by minimizing a prediction error function between the current and the previous frame, known as the *displaced frame difference* function. In the multiple-frame formulation the velocity field vector is the

---

[1] This material is based on work supported in part by NATO under Grant No. 0103/88.

one which simultaneously minimizes ν prediction error functions, each one between the ν-th previous frame and the current one. The Wiener-based algorithm in [1] is a special case of the algorithm discussed here with $\nu = 1$. The algorithms of Cafforio and Rocca and Walker and Rao [3] are also related to the algorithms derived from a Wiener-based formulation [1]. The Wiener-based algorithms give a very stable and fast converging displacement estimation by requiring only a small increase of the computational load using causal spatial mask rather than one point. The multiple-frame algorithm, however, requires the storage of ν frames and it is computationally more expensive.

In Sec. 2 the multiple frame formulation is presented and terminology is established. In the same section, the derivation of two motion estimation algorithms is presented. In Sec. 3 various important implementation issues are discussed, such as, a spatially adaptive regularization approach and a prediction model for the assignment of the initial estiate of the motion vectors. In Sec. 4 we present a number of experiments, in which the performance of the multiple-frame algorithms is compared to the two frame algorithm of [1]. The reduction of the prediction error and the smoothness of the estimated velocity field are two issues of interest. The proposed algorithms have a significantly reduced prediction error when compared with the methods proposed by Walker and Rao and by Cafforio and Rocca, as well as with the well-known steepest-descent approach [8]. The performance of the proposed algorithm in the presence of noise is studied. Finally, the conclusions drawn from our experiments and future research topics are covered in Sec. 5.

2 Derivation of the Multiple-Frame Wiener-based Algorithms

2.1 Multiple-Frame Formulation

The basic idea behind the two-frame pel-recursive algorithms is the following: if a pixel intensity has changed between two subsequent frames, it is assumed to belong to a moving area. The aim is to find the intensity value in the previous frame which corresponds to the current working point and, consequently, the displacement vector. In formulating the problem using more than one previous frame we need to consider the intensity changes between the current frame and each of the previous available frames.

More specifically, let f_k represent the intensity values at current k-th frame and assume that ν previous frames, namely, f_{k-i}, $i = 1, \ldots \nu$, are available. Let us denote by $f_{k-i}(\mathbf{r}_{k-i})$, $i = 0, \ldots \nu$, the intensity value at any given pixel which is located at the position $\mathbf{r}_{k-i}$ at the $(k-i)$-th frame. Vector $\mathbf{r}_k$ denotes the (m, n) point on the rectangular raster of the current frame $f_k(m, n)$, $1 \leq m \leq M$, $1 \leq n \leq N$. Furthermore, let us denote the displacement vector from frame f_{k-i} to the point $\mathbf{r}_{k-j}$ of frame f_{k-j} by $\mathbf{d}_{k-i,k-j}(\mathbf{r}_{k-j})$, where $\mathbf{r}_{k-j} = \mathbf{r} - \mathbf{d}_{k-j,k}(\mathbf{r})$ for $0 \leq j < i \leq \nu$ and $\mathbf{r} \equiv \mathbf{r}_k$. In the following vector $\mathbf{r}_k$ will be denoted by $\mathbf{r}$ and all vectors $\mathbf{d}_{k-j,k}(\mathbf{r})$, $j = 1, \ldots \nu$, that is, the vectors pointing to the current frame f_k, will be denoted as $\mathbf{d}_{k-j,k}$, for simplicity. With the above notation, the motion vector from frame f_{k-i} to frame f_k is equal to

$$\begin{aligned} \mathbf{d}_{k-i,k} &= \sum_{j=1}^{i} \mathbf{d}_{k-j,k-j+1}(\mathbf{r} - \mathbf{d}_{k-j+1,k}) \\ &= \mathbf{d}_{k-i,k-1}(\mathbf{r} - \mathbf{d}_{k-1,k}) + \mathbf{d}_{k-1,k}, \end{aligned} \tag{1}$$

where $\mathbf{d}_{k,k} \equiv \mathbf{0}$. Our goal is to find an estimate $\hat{\mathbf{d}}_{k-1,k}$ of the true motion vector $\mathbf{d}_{k-1,k}$, given the peviously $\nu - 1$ estimated vectors $\hat{\mathbf{d}}_{k-i,k-i+1}(\mathbf{r} - \hat{\mathbf{d}}_{k-i+1,k})$, $i = 2, \ldots \nu$. Towards this point, for any vector $\bar{\mathbf{d}}_{k-i,k}$ let us define the *i-th displaced frame difference* as follows

$$\Delta^{(i)}(\mathbf{r}, \bar{\mathbf{d}}_{k-i,k}) = f_k(\mathbf{r}) - f_{k-i}(\mathbf{r} - \bar{\mathbf{d}}_{k-i,k}). \tag{2}$$

An estimate $\hat{\mathbf{d}}_{k-i,k}$ of $\mathbf{d}_{k-i,k}$ can be obtained by minimizing $\Delta^{(i)}(\mathbf{r}, \bar{\mathbf{d}}_{k-i,k})$, that is,

$$\hat{\mathbf{d}}_{k-i,k} = \arg[\min_{\bar{\mathbf{d}}_{k-i,k}} \Delta^{(i)}(\mathbf{r}, \bar{\mathbf{d}}_{k-i,k})]. \tag{3}$$

Let $\mathbf{d}^{\circ}_{k-i,k}$ be an initial estimate of $\mathbf{d}_{k-i,k}$. Then, based on the commonly used assumption that knowledge of the true motion vectors allows the *perfect registration* of the frames, that is,

$$f_k(\mathbf{r}) = f_{k-i}(\mathbf{r} - \mathbf{d}_{k-i,k}), \tag{4}$$

Eq. (2) for $\bar{\mathbf{d}}_{k-i,k} = \mathbf{d}^{\circ}_{k-i,k}$ becomes

$$\Delta^{(i)}(\mathbf{r}, \mathbf{d}^{\circ}_{k-i,k}) = f_{k-i}(\mathbf{r} - \mathbf{d}_{k-i,k}) - f_{k-i}(\mathbf{r} - \mathbf{d}^{\circ}_{k-i,k}). \tag{5}$$

Linearizing $\Delta^{(i)}$ by using the Taylor series expansion of the intensity function $f_{k-i}(\mathbf{r} - \mathbf{d}_{k-i,k})$, at the location $(\mathbf{r} - \mathbf{d}^{\circ}_{k-i,k})$, we get

$$f_{k-i}(\mathbf{r} - \mathbf{d}_{k-i,k}) = f_{k-i}(\mathbf{r} - \mathbf{d}^{\circ}_{k-i,k}) - (\mathbf{d}_{k-i,k} - \mathbf{d}^{\circ}_{k-i,k})^T \nabla f_{k-i}(\mathbf{r} - \mathbf{d}^{\circ}_{k-i,k}) + v(\mathbf{r}, \mathbf{d}^{\circ}_{k-i,k}), \tag{6}$$

where ^{T} denotes the transpose of a vector or a matrix, $\nabla f(\cdot)$ represents the two-dimensional gradient operator and the term $v(\mathbf{r}, \mathbf{d}^{\circ}_{k-i,k})$ equals the truncation error resulting from the linearization. If $\mathbf{d}^{\circ}_{k-1,k}$ is an initial estimate of $\mathbf{d}_{k-1,k}$, then according to (1)

$$\mathbf{d}^{\circ}_{k-i,k} = \mathbf{d}_{k-i,k-1}(\mathbf{r} - \mathbf{d}^{\circ}_{k-1,k}) + \mathbf{d}^{\circ}_{k-1,k}. \tag{7}$$

Assuming that the estimated displacement vector $\hat{\mathbf{d}}_{k-i,k-1}$ is equal to the true vector $\mathbf{d}_{k-i,k-1}$, and consequently

$$\mathbf{d}_{k-i,k-1}(\mathbf{r} - \mathbf{d}^{\circ}_{k-1,k}) = \hat{\mathbf{d}}_{k-i,k-1}(\mathbf{r} - \mathbf{d}^{\circ}_{k-1,k}), \tag{8}$$

from (1) and (7) we get

$$\mathbf{d}_{k-i,k} - \mathbf{d}^{\circ}_{k-i,k} = \mathbf{d}_{k-1,k} - \mathbf{d}^{\circ}_{k-1,k}. \tag{9}$$

By substitution of (6) and (9) into (5) we obtain the following expression

$$\Delta^{(i)}(\mathbf{r}, \mathbf{d}^{\circ}_{k-i,k}) = -\nabla^T f_{k-i}(\mathbf{r} - \mathbf{d}^{\circ}_{k-i,k}) \cdot (\mathbf{d}_{k-1,k} - \mathbf{d}^{\circ}_{k-1,k}) + v(\mathbf{r}, \mathbf{d}^{\circ}_{k-i,k}). \tag{10}$$

The truncation error term $v(\mathbf{r}, \mathbf{d}^{\circ}_{k-i,k})$ is not neglected but instead it is considered to be a sample of a stochastic process, as was done in [1, 4]. Furthermore, we assume that the true update $\mathbf{u} = \mathbf{d}_{k-1,k} - \mathbf{d}^{\circ}_{k-1,k}$ is a sample of a stochastic process as well. Our objective now is to find a linear least-squares estimate of the update term $\mathbf{u}$, which gives correction to the initial estimate $\mathbf{d}^{\circ}_{k-1,k}$. For this purpose, we consider the displaced frame difference terms in (10) as an observation and the gradient as a known vector.

A solution to the estimation problem could be obtained using information of the current pixel and its corresponding points on the ν previous frames. However, if the current pixel belongs to a moving area, then all pixels in a small pre-specified neighborhood are used to find a more stable displacement estimation. For the general case of w observations, a Wiener-based estimation procedure of the update is discussed in the next subsection.

2.2 A Multiple-Frame Wiener Approach

In this section, a linear least-squares estimate of the update is derived based on w observation points $\mathbf{r}_j$, $j = 1, 2, \dots w$, belonging to a spatial mask $\mathcal{W}$ on frame f_k, and their corresponding pixels in the previous ν frames. More specifically, using (7) and the observations on frames f_k and f_{k-i}, we obtain

$$\begin{bmatrix} \Delta^{(i)}(\mathbf{r}_1, \mathbf{d}^{\circ}_{k-i,k}) \\ \vdots \\ \Delta^{(i)}(\mathbf{r}_w, \mathbf{d}^{\circ}_{k-i,k}) \end{bmatrix} = - \begin{bmatrix} g_x^{(i)}(1) & g_y^{(i)}(1) \\ \vdots & \vdots \\ g_x^{(i)}(w) & g_y^{(i)}(w) \end{bmatrix} \cdot [\mathbf{d}_{k-1,k} - \mathbf{d}^{\circ}_{k-1,k}] + \begin{bmatrix} v(\mathbf{r}_1, \mathbf{d}^{\circ}_{k-i,k}) \\ \vdots \\ v(\mathbf{r}_w, \mathbf{d}^{\circ}_{k-i,k}) \end{bmatrix}, \tag{11}$$

where $g_x^{(i)}(j)$ and $g_y^{(i)}(j)$ denote the gradients in the x and y directions, respectively, at the spatial location $\mathbf{r}_j - \mathbf{d}^{\circ}_{k-i,k}$ at frame f_{k-i}.

Alternatively, by using vectors and matrices, Eq. (11) is written as

$$\mathbf{z}^{(i)} = G^{(i)}(\mathbf{d}_{k-1,k} - \mathbf{d}^{\circ}_{k-1,k}) + \mathbf{v}^{(i)} = G^{(i)}\mathbf{u} + \mathbf{v}^{(i)}, \tag{12}$$

where $\mathbf{z}^{(i)}$ and $\mathbf{v}^{(i)}$ are $w\times 1$ vectors and $G^{(i)}$ is a known $w\times 2$ transition matrix of gradients. Using this model, we want to find a linear least-squares estimate $\hat{\mathbf{u}}$ of $\mathbf{u}$, given $\mathbf{z}^{(i)}$ and $G^{(i)}$. From Eq. (12) it is clear that, by using the observations at the current frame f_k and those on any of the available frames f_{k-i}, we can get ν different estimates, $\hat{\mathbf{u}}^{(i)}$, $i = 1, \ldots \nu$, of the update $\mathbf{u}$. One way to combine all these estimates is given below. It requires the assumption we made earlier, that is, the previously estimated displacement vectors are equal to the true ones. In other words, under the assumption of perfect registration of the frames, the temporal correlation coefficient of the (noise-free) observations in the direction of the motion is equal to one. However, the above is not true in practice and, therefore, the registration error has to be considered in the estimation algorithm. One way to do that is to weight the ν estimates of $\mathbf{u}$ differently, that is, introduce weights w_i, $i = 1, \ldots \nu$, and obtain a weighted least-squares solution. Let us define the vectors $\mathbf{z}$, $\mathbf{v}$ and the matrix G by stacking $\mathbf{z}^{(i)}$, $\mathbf{v}^{(i)}$ and $G^{(i)}$. Then, the following system of equations is obtained

$$\mathbf{z} = G(\mathbf{d}_{k-1,k} - \mathbf{d}^{\circ}_{k-1,k}) + \mathbf{v} = G\mathbf{u} + \mathbf{v}, \tag{13}$$

where the vectors $\mathbf{z}$ and $\mathbf{v}$ are of size $\nu w\times 1$ and G is an $\nu w\times 2$ matrix.

In obtaining a linear least-squares estimate of $\mathbf{u}$, it is assumed that $\mathbf{u}$ and $\mathbf{v}$ are zero mean uncorrelated Gaussian processes with covariance matrices R_u and R_v, respectively [1]. However, here a weighted least-squares approach is followed by introducing a block-diagonal weight matrix W of size $\nu w\times\nu w$. Each of the ν blocks of matrix W is a diagonal matrix of size $w\times w$ with all its diagonal elements equal to w_i, $i = 1, \ldots \nu$, such that $\sum_{i=1}^{\nu} \mathrm{w}_i^2 = 1$. The different weighting in obtaining the solution is justified from the fact that the error due to misregistration of previous frames, increases for frames away from the current one, thus, their contribution to the solution should be smaller. Then, an estimate $\hat{\mathbf{u}}$ of $\mathbf{u}$ is given by

$$\hat{\mathbf{u}} = \left[G^T W^T R_v^{-1} W G + R_u^{-1}\right]^{-1} \cdot G^T W^T R_v^{-1} W \mathbf{z}. \tag{14}$$

Furthermore, due to lack of a priori knowledge about the covariance matrices, we simplify the above expression. We assume that the corresponding covariance matrices may be written as $R_u = \sigma_u^2 \cdot I$ and $R_v = \sigma_v^2 \cdot I$. Then, $\hat{\mathbf{u}}$ is equal to

$$\hat{\mathbf{u}} = [G^T W^T W G + \mu \cdot I]^{-1} \cdot G^T W^T W \mathbf{z} \tag{15}$$

or

$$\hat{\mathbf{u}} = -\begin{bmatrix} \sum_{i=1}^{\nu} \mathrm{w}_i^2 \sum_{j=1}^{w} g_x^{(i)2}(j) + \mu & \sum_{i=1}^{\nu} \mathrm{w}_i^2 \sum_{j=1}^{w} g_x^{(i)}(j) g_y^{(i)}(j) \\ \sum_{i=1}^{\nu} \mathrm{w}_i^2 \sum_{j=1}^{w} g_x^{(i)}(j) g_y^{(i)}(j) & \sum_{i=1}^{\nu} \mathrm{w}_i^2 \sum_{j=1}^{w} g_y^{(i)2}(j) + \mu \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_{i=1}^{\nu} \mathrm{w}_i^2 \sum_{j=1}^{w} g_x^{(i)}(j) \cdot \Delta^{(i)}(\mathbf{r}_j, \mathbf{d}^{\circ}_{k-i,k}) \\ \sum_{i=1}^{\nu} \mathrm{w}_i^2 \sum_{j=1}^{w} g_y^{(i)}(j) \cdot \Delta^{(i)}(\mathbf{r}_j, \mathbf{d}^{\circ}_{k-i,k}) \end{bmatrix}, \tag{16}$$

where the regularization parameter μ can be set equal to σ_v^2/σ_u^2 or it can be evaluated as discussed in Sec. 3. Then, the estimate of $\mathbf{d}_{k-i,k}$ at any spatial location is given by

$$\hat{\mathbf{d}}_{k-1,k} = \mathbf{d}^{\circ}_{k-1,k} + \hat{\mathbf{u}}. \tag{17}$$

The above procedure can be repeated by considering the newly obtained estimate as an initial estimate for computing a new update and so on. However, as was shown in [1], at least for the two-frame Wiener-based algorithm, one iteration is enough to obtain very satisfactory results. In our work we also arrived at a similar conclusion.

2.3 A Model-Based Wiener Approach

The algorithm in the previous section was derived by assuming that Eq. (4) holds for any previous frame and by assigning weights to compensate for the error due to misregistration of the frames. In this section we derive a multiple frame algorithm following on a model based approach. In general, Eq. (4) does not hold true for the entire frame due primarily to occlusion effects occurring when moving objects are present. Furthermore, when only estimates of the true vectors are available the correlation coefficient of the registered frames is less than one. We assume the following AR model for the registered frames

$$f_k(\mathbf{r}) = \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot f_{k-i}(\mathbf{r} - \mathbf{s} - \mathbf{d}_{k-i,k}) + \theta(\mathbf{r}), \tag{18}$$

where $\mathbf{r} = (m,n)$, $\mathbf{s} = (p,q)$, $a_i(\mathbf{s})$ are the prediction coefficients and $\mathcal{S}$ is the 3-D support. The term $\theta(\mathbf{r})$ is a zero-mean white noise process with variance equal to the mean-squared prediction error ϵ^2, that is, $\epsilon^2 = E\{\theta^2(\mathbf{r})\}$. The prediction coefficients can be computed by solving the following set of *normal equations*, that is,

$$R(\bar{\mathbf{s}},\bar{i}) - \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot R(\bar{\mathbf{s}} - \mathbf{s}, \bar{i} - i) = 0, \qquad (\bar{\mathbf{s}},\bar{i})\in\mathcal{S}, \tag{19}$$

where $\bar{\mathbf{s}} = (\bar{p},\bar{q})$ and

$$R(\bar{\mathbf{s}},\bar{i}) = E\{f_k(\mathbf{r}) f_{k-\bar{i}}(\mathbf{r} - \bar{\mathbf{s}} - \mathbf{d}_{k-\bar{i},k})\}. \tag{20}$$

In formulating the problem using two frames, we can assume that the following simple correlation model describes the registered frames, that is,

$$R(\bar{p},\bar{q},\bar{i}) = \sigma_s^2 \rho_v^{|\bar{p}|} \rho_h^{|\bar{q}|} \rho_t^{|\bar{i}|}, \tag{21}$$

where ρ_v, ρ_h and ρ_t are the correlation coefficients in the vertical, horizontal and trajectory of motion directions, respectively, and σ_s^2 is the signal variance. The causal support $\mathcal{S}$ is shown in Fig. 1, and the mean squared prediction error in this case is shown to be equal to

$$\epsilon^2 = (1-\rho_h^2)(1-\rho_v^2)(1-\rho_t^2)\sigma_s^2. \tag{22}$$

Now, by considering any arbitrary vector $\bar{\mathbf{d}}_{k-1,k}$, let us define the *predicted displaced frame difference* as follows

$$\Delta(\mathbf{r},\bar{\mathbf{d}}_{k-1,k}) = f_k(\mathbf{r}) - \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot f_{k-i}(\mathbf{r} - \mathbf{s} - \bar{\mathbf{d}}_{k-i,k}), \tag{23}$$

which can be considered as a prediction error function, where the prediction is done by both the AR model and by assigning to $\bar{\mathbf{d}}_{k-1,k}$ an initial estimate of $\mathbf{d}_{k-1,k}$. Then, an estimate of the true vector $\mathbf{d}_{k-1,k}$ can be obtained by minimizing Δ with respect to $\bar{\mathbf{d}}_{k-1,k}$.

By substituting the expansion of $f_{k-i}(\mathbf{r}-\mathbf{s}-\mathbf{d}_{k-i,k})$ similar to that of Eq. (6) into Eq. (18) and by considering Eq. (9), we get

$$\begin{aligned} f_k(\mathbf{r}) - \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot f_{k-i}(\mathbf{r}-\mathbf{s}-\mathbf{d}^{\circ}_{k-i,k}) &= -(\mathbf{d}_{k-1,k}-\mathbf{d}^{\circ}_{k-1,k})^T \\ \cdot \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot\nabla f_{k-i}(\mathbf{r}-\mathbf{s}-\mathbf{d}^{\circ}_{k-i,k}) &+ \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot v(\mathbf{r}-\mathbf{s}-\mathbf{d}^{\circ}_{k-i,k}) + \theta(\mathbf{r}), \end{aligned} \tag{24}$$

or

$$\Delta(\mathbf{r},\mathbf{d}^{\circ}_{k-1,k}) = -\left\{ \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot\nabla^T f_{k-i}(\mathbf{r}-\mathbf{s}-\mathbf{d}^{\circ}_{k-i,k}) \right\} \cdot (\mathbf{d}_{k-1,k}-\mathbf{d}^{\circ}_{k-1,k}) + \eta(\mathbf{r},\mathbf{d}^{\circ}_{k-1,k}), \tag{25}$$

where $\eta(\mathbf{r},\mathbf{d}^{\circ}_{k-1,k})$ denotes the sum of the error terms at the right-hand side of Eq. (24). Based on Eq. (25) and using a spatial window $\mathcal{W}$, as in Sec. 2.2, a system of equations can be written and then solved for an estimate of the update $\mathbf{u}$. In following a linear least-squares approach, the statistics of $\eta(\bar{r},\mathbf{d}^{\circ}_{k-1,k})$ are required. Using the assumption that $R_v = \sigma_v^2\cdot I$ and that the linearization error $v(\cdot)$ and the prediction error $\theta(\mathbf{r})$ are uncorrelated stochastic processes, we get that $R_\eta = \sigma_\eta^2\cdot I$, where

$$\sigma_\eta^2 = \sigma_v^2 \cdot \sum_{(\mathbf{s},i)\in\mathcal{S}} a_i^2(\mathbf{s}) + \epsilon^2. \tag{26}$$

The Wiener-based estimate of the update is given now by

$$\hat{\mathbf{u}} = -\begin{bmatrix} \sum_{j=1}^{w} \hat{g}_x^2(j) + \mu & \sum_{j=1}^{w} \hat{g}_x(j)\hat{g}_y(j) \\ \sum_{j=1}^{w} \hat{g}_x(j)\hat{g}_y(j) & \sum_{j=1}^{w} \hat{g}_y^2(j) + \mu \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_{j=1}^{w} \hat{g}_x(j)\cdot\Delta(\mathbf{r}_j,\mathbf{d}^{\circ}_{k-1,k}) \\ \sum_{j=1}^{w} \hat{g}_y(j)\cdot\Delta(\mathbf{r}_j,\mathbf{d}^{\circ}_{k-1,k}) \end{bmatrix}, \tag{27}$$

where

$$\sum_{(\mathbf{s},i)\in\mathcal{S}} a_i(\mathbf{s})\cdot\nabla f_{k-i}(\mathbf{r}_j-\mathbf{s}-\mathbf{d}^{\circ}_{k-i,k}) = [\hat{g}_x(j)\ \ \hat{g}_y(j)]^T, \tag{28}$$

and the regularization parameter μ can be set equal to σ_η^2/σ_u^2 or it can be evaluated as discussed in Sec. 3. The estimate of $\mathbf{d}_{k-1,k}$ is given by Eq. (17).

3 Various Implementation Issues

The value of the regularization constant μ is important for the performance of the algorithm and was kept constant in previous work [1, 4]. For small values of μ the pel-recursive algorithm is quite accurate but at the same time very sensitive in the presence of noise. On the other hand, a large value of μ can make the algorithm less accurate in detecting displacements of small objects. In our work we adapt the value of μ. This adaptation is based on the fact that the pel-recursive algorithm is more sensitive in the near edge areas of moving objects where occlusion occurs, due primarily to the ill-conditionedness of G. Thus, in order to improve the performance of the algorithm we propose to set μ, which serves as a regularization parameter, proportional to the ratio of the two singular values of G^TG. For small ratios μ is set to a constant threshold. On the other hand, for very large ratios we set the update equal to zero. This adaptive regularization of the inversion of (19) has improved the estimation results.

The choice of the initial estimate $\mathbf{d}^{\circ}_{k-1,k}$ at each point (m,n) is very important, since it determines the number of iterations required in obtaining an accurate estimate of the motion vector. One choice is to use $\mathbf{d}^{\circ}_{k-1,k}(m,n) \doteq \hat{\mathbf{d}}_{k-1,k}(m,n-1)$, as was done in [1]. We propose the modeling of the motion vector field by a 3-D AR model of a causal support $\mathcal{S}_3$, that is,

$$\mathbf{d}^{\circ}_{k-1,k}(\mathbf{r}) = \sum_{(\mathbf{s},i)\in\mathcal{S}_3} \gamma_i(\mathbf{s})\cdot\hat{\mathbf{d}}_{k-i-1,k-i}(\mathbf{r}-\mathbf{s}-\bar{\mathbf{d}}^{\circ}_{k-i,k}), \tag{29}$$

where $\mathbf{r} = (m,n)$, $\mathbf{s} = (p,q)$ and $\gamma_r(\mathbf{s})$ are the prediction coefficients. The vector $\bar{\mathbf{d}}^{\circ}_{k-i,k}$ is an initial estimate of $\mathbf{d}_{k-i,k}$, with $\bar{\mathbf{d}}^{\circ}_{k-1,k}$ computed based on prediction using a spatial support, that is,

$$\bar{\mathbf{d}}^{\circ}_{k-1,k}(m,n) = \sum_{(p,q)\in\mathcal{S}_2} \gamma(p,q)\cdot\hat{\mathbf{d}}_{k-1,k}(m-p,n-q), \tag{30}$$

where $\mathcal{S}_2$ is a causal spatial support. A simple form of Eq. (30) was used in [5] with $\gamma(1,0) = \gamma(0,1) = 0.5$ and $\mathbf{d}^{\circ}_{k-1,k}$ was set equal to $\bar{\mathbf{d}}_{k-1,k}$. However, Eq. (29) is justified since the displacement vectors can be as much correlated in the temporal direction as they are in the spatial directions. According to our experimental evidence, the above approach is shown to be very effective, especially when large moving objects are present or in head-and-shoulders video-conference scenes. In our work, a simple 3-D support was used with $\gamma_0(1,0) = \gamma_0(0,1) = 0.3$ and $\gamma_1(0,0) = 0.4$.

In implementing the algorithm, a threshold detector was used to detect the motion, as done in [1, 4]. The update was set equal to zero at the locations where the inequality

$$\sum_{\mathbf{r}\in\mathcal{W}} |\Delta(\mathbf{r},\mathbf{d}^{\circ}_{k-1,k})| < T \tag{31}$$

was satisfied, where T is the threshold value and Δ is the first displaced frame difference. On the other hand, at the locations where the inequality

$$\sum_{\mathbf{r}\in\mathcal{W}} |f_k(\mathbf{r}) - f_{k-1}(\mathbf{r})| < \sum_{\mathbf{r}\in\mathcal{W}} |\Delta(\mathbf{r},\mathbf{d}^{\circ}_{k-1,k})| \tag{32}$$

was satisfied, the vector $\hat{\mathbf{d}}_{k-1,k}$ was set equal to zero Finally, in order to evaluate the intensity value and the displacement vector at an arbitrary position in the images, bilinear interpolation was used. Figure 1 shows the implementation of the summation of Eq. (1).

4 Experimental Results

In this section a number of experimental results are presented. The algorithm was tested on the standard video-conference monochromatic 8-bit image sequence "Trevor White" of size 256×256×150. In addition, an artificially composed image sequence called "T.W.Head", of size 128×128×25 was used. The head of the speaker from the "Trevor White" sequence was extracted and was displaced by two pixels to the right at each frame in a stationary background. This image sequence proved to be very useful for the comparison of the motion estimation algorithms since the motion was exactly known.

The algorithms were compared with respect to their *improvement in motion compensation* at each frame, which is defined in *decibel* (dB) by

$$I_{MC}(k) = 10\cdot\log_{10}\left\{\frac{\|f_k(\mathbf{r}) - f_{k-1}(\mathbf{r})\|^2}{\|f_k(\mathbf{r}) - f_{k-1}(\mathbf{r}-\hat{\mathbf{d}}_{k-1,k})\|^2}\right\}. \tag{33}$$

A second figure of merit used was based on the entire sequence of K frames. The average improvement in motion compensating is defined by

$$\bar{I}_{MC} = 10\cdot\log_{10}\left\{\frac{\sum_{k=2}^{K}\|f_k(\mathbf{r}) - f_{k-1}(\mathbf{r})\|^2}{\sum_{k=2}^{K}\|f_k(\mathbf{r}) - f_{k-1}(\mathbf{r}-\hat{\mathbf{d}}_{k-1,k})\|^2}\right\}. \tag{34}$$

Figure 2 shows $I_{MC}(k)$ for the frames 22 to 40 of the "Trevor White" sequence using the algorithms presented above. Curves a, b, c correspond to the algorithm of Sec. 2.2, for $\nu = 1, 2, 3$, and curve d corresponds to the algorithm of Sec. 2.3, where the simple model of Eq. (21) with a two-frame causal support of eight points was used. The corresponding $\bar{I}_{MC}$'s are equal to 5.944, 6.963, 6.916 and 6.411 dB. The multiple-frame pel-recursive algorithm for $\nu = 2, 3$ performs better than the two-frame one ($\nu = 1$). However, the algorithm for $\nu = 3$ does performs better than the algorithm for $\nu = 2$. An explanation for this behavior is that errors are introduced in the estimate of the $\mathbf{d}_{k-1,k}$ vector due to the assumptions made in the derivation of the algorithm. Finally, the model-based two-frame algorithm has shown a significant reduction in the prediction error compared to the standard pel-recursive algorithm, thus justifying the importance of the new approach.

The algorithms were also tested in the presence of noise. The above image sequences were distorted by adding white Gaussian zero-mean noise to them. For low *SNR*'s a spatial 2-D recursive filter, referred to as Kak's point-to-point estimator [9], can be used to smooth the images before the pel-recursive algorithms are applied. However, the objective of this experiment is the determination of the robustness of the algorithms in estimating a smooth velocity field, in the presence of noise. In order to measure the performance of the pel-recursive algorithms in the presence of noise, we use the improvement in motion compensation with respect to the original noise-free image sequence, which is available in a simulation environment. This improvement, denoted $I^{\circ}_{MC}(k)$ and $\bar{I}^{\circ}_{MC}$, is given by (25) and (26), respectively, where the frames f_k are substituted by the original noise-free ones. Figure 3 shows $I^{\circ}_{MC}(k)$ for the "T.W.Head" sequence and $SNR = 20$ dB. Curves a,b correspond to the algorithm of Sec. 2.2, for $\nu = 1, 2$, and curve c corresponds to the algorithm of Sec. 2.3, The corresponding $\bar{I}^{\circ}_{MC}$'s are equal to 9.98, 10.84 and 10.55 dB. Clearly, both the multiple-frame algorithm ($\nu = 2$) and the model-based algorithm are more effective in estimating the true motion field. Such a motion field is shown in Figs. 4a,b. Figure 4a shows the velocity field obtained with the use of the two frame pel-recursive algorithm, implemented according to discussion in Sec. 3. The velocity field for the same frame obtained with the use of the multiple-frame pel-recursive algorithm with $\nu = 2$ is shown in Fig. 4b. By comparing the two figures it is clear that the motion field in Fig. 4b is smoother that the motion field in Fig. 4a.

5 Conclusions and Future Research

The problem of motion estimation in moving images is undoubtedly very important in many applications. In this paper a multiple-frame formulation of the pel-recursive motion estimation algorithm was presented using both the standard and a model-based approach. The pel-recursive algorithms that appeared in the literature are special cases of the algorithm presented here for $\nu = 1$. The algorithm for $\nu > 1$ is computationally more expensive and requires storage of ν frames in the coder. However, for any ν it requires the inversion of a 2×2 matrix and the implementation requirements in the decoder is the same, since only the motion vectors $\hat{\mathbf{d}}_{k-1,k}(m,n)$ need to be transmitted. On the other hand, the proposed algorithm has a number of advantages. One of these advantages is that it is less sensitive to noise and the solution is more stable, since ν times more points are used for the same mask size w. Furthermore, the performance of the algorithm is improved without increasing the number of samples (mask size) in the current frame, since such an increase causes the pel-recursive algorithm to be less accurate in near edge areas and less robust when objects are occluded. Experimental results with standard video sequences confirm that the multiple frame motion estimation algorithm outperforms the two-frame based algorithm and other pel-recursive techniques on the basis of robustness, stability and smoothness of the motion field.

It is probably safe to say that, the solution to the general motion estimation problem based on more than just two frames is yet in its infancy. Therefore, our attention will be focused in the formulation of the problem using various image and motion models in order to improve the already obtained results. As was indicated by the experimental results, it is important that the existence of noise is directly incorporated into the problem formulation. Thus, there is a need to develop algorithms which take advantage of the temporal correlation in order to achieve noise filtering and improve the motion estimation, simultaneously.

References

[1] J. Biemond, L. Looijenga and D.E. Boekee, "A Pel-Recursive Wiener-Based Displacement Estimation Algorithm," *Signal Processing 13*, pp. 399-412 (1987).

[2] H.G. Mushmann, P. Pirsh and H.J. Grallert, "Advances in Picture Coding," *Proc. IEEE,* vol. 73, no. 4, pp. 523-548 (1985).

[3] D.R. Walker and K.R. Rao, "Improved Pel-Recursive Motion Estimation," *IEEE Trans. Commun.,* vol. COM-32, no. 10, pp. 1128-1134 (1984).

[4] J. Biemond, J.N. Driessen, A.M. Geurtz and D.E. Boekee, "A Pel-Recursive Wiener-Based Algorithm for the Simultaneous Estimation of Rotation and Translation," *Proc. SPIE Conf. Visual Commun. and Image Processing*, vol. 1001, pp. 917-924 (1988).

[5] J.N. Driessen, J. Biemond and D.E. Boekee, "A Pel-Recursive Segmentation and Estimation Algorithm for Motion Compensated Image Sequence Coding," *Proc. 1989 Int. Conf. Acoust., Speech, Signal Processing*, pp. 1901-1904, Glasgow, Scotland (1989).

[6] A.K. Katsaggelos, J. Driessen, S.N. Efstratiadis and R.L. Lagendijk, "Spatio-Temporal Motion Compensated Noise Filtering of Image Sequences," *Proc. SPIE Conf. Visual Commun. and Image Processing*, vol. 1199, Philadelphia, PA, November (1989).

[7] F.W. Mounts, "A Video Encoding System with Conditional Pel-element Replenishment," *The Bell Syst. Techn. Journ.*, vol. 58, no. 3, pp. 1735-1745 (1980)

[8] A.N. Netravali and J.D. Robins, "Motion-Compensated Television Coding: Some New Results," *The Bell Syst. Techn. Journ.*, vol. 59, no. 9, pp. 1735-1745 (1980).

[9] A. Rosenfeld and A.C. Kak, *Digital Picture Processing,* 2nd edition, Academic Press (1982).

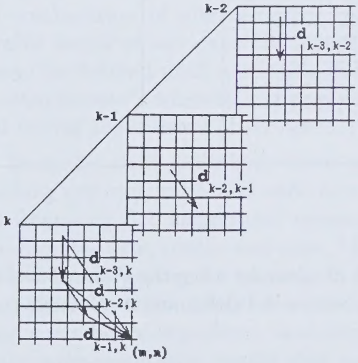


Fig. 1. Addition of the displacement vectors.

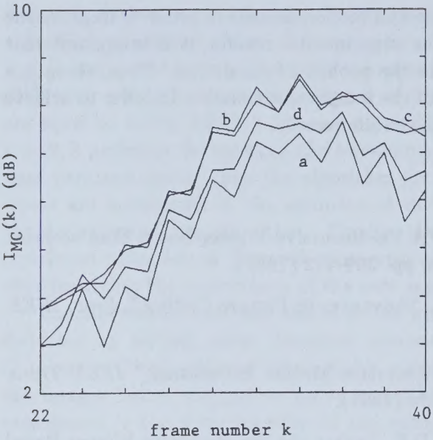


Fig. 2. $I_{MC}(k)$ for the algorithms of Sec. 2.2 for $\nu = 1, 2, 3$ (a,b,c) and Sec. 2.3 for $\nu = 1$ (d).

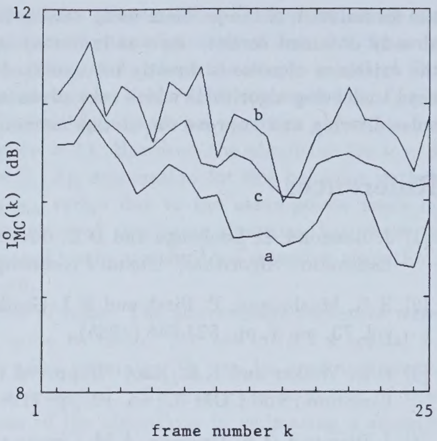


Fig. 3. $I^{o}_{MC}(k)$ for the algorithms of Sec. 2.2 for $\nu = 1, 2$ (a,b) and Sec. 2.3 for $\nu = 1$ (c).

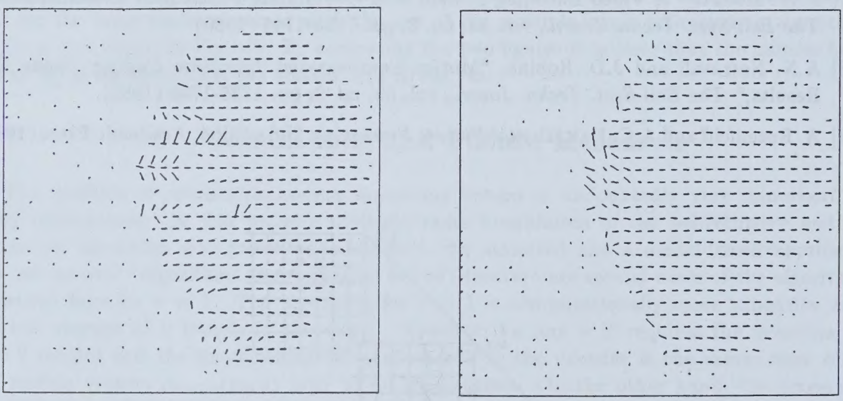


Fig. 4. Velocity fields obtained by using the Wiener-based algorithm of Sec. 2.2 with $\nu = 1$ (left), and $\nu = 2$ (right).

EXACTLY SOLVABLE TELEPHONE SWITCHING PROBLEMS

NICHOLAS PIPPENGER
University of British Columbia, Vancouver, BC, Canada

Abstract: For a certain class of telephone switching problems, much of our understanding arises from an analogy with statistical mechanics that was proposed by Beneš in 1963. This analogy has lead to the exact solution of a number of idealised problems, which we survey in this paper.

1. INTRODUCTION

This paper aims to survey a certain class of results about a certain class of communication problems. The class of problems may be described as follows.

Streams of requests, of one or more different types, arrive at a system for service. The streams are independent Poisson random processes with stationary mean interarrival times that depend only on the type of request. A request of a given type can be satisfied by one or more alternative resources or combinations of resources within the system. At each moment in time, each resource is either idle or busy. If a suitable combination of idle resources is available when a request arrives, such a combination is engaged by the request, and the constituent resources become busy for the interval of time required to service the request. The durations of these service intervals are independent exponentially distributed random variables, with means that depend only on the type of request. At the end of the service interval, the engaged resources become idle, and are once again available to serve other requests. If no suitable combination of resources is idle when a request arrives, the request is lost (not served) and has no further effect on the system.

Various problems in this class arise according to various specifications of the types of requests and their arrival rates and service times, the types of resources in the system and the combinations suitable to the different types of requests, and the policy according to which a particular suitable combination of idle resources is chosen when more than one is available for a request. Variations that do *not* fall within the scope of this class are (1) "refusals", whereby a request may be denied service despite the availability of a suitable combination of idle resources (presumably in the hope of preventing even less desirable denial of service in the future), (2) "deferrals", whereby the service of a request may be delayed until a suitable combination of idle resources is available, and (3) "rearrangements", whereby the combination of resources serving a request may be changed during the interval of service.

Numerous problems from the class described above are met in the design and analysis of telephone switching systems, local- and wide-area computer communication networks, and time- and frequency-slot assignment systems. All of these problems reduce to the analysis of a discrete-state, continuous-time, Markov jump process which, if the number of states is not too large, can be carried out by standard methods. For many problems of interest, however, the number of states is too large to allow such a solution by brute force, and a number of expedients have been devised to deal with such problems. Firstly, there are some qualitative results that apply to all problems in the class (and thus of necessity give rather little information about any particular problem).

For example, a theorem of Beneš [Be66] says that, if we seek to minimise the rate at which requests are lost, we may confine our search to policies that are "deterministic", in the sense that, to a given type of request arriving in a given state, they always assign the same combination of resources (rather than randomising among the available combinations). Secondly, there are methods of approximation by which one may hope to obtain rough quantitative information about a complicated situation. For example, there is a vast literature devoted to the analysis of "channel graphs", which at its outset replaces the probability distribution on the states by one in which each resource in idle or busy independently of the others. (For a survey of this literature as of 1979, see Hwang [Hw79]. For a more general account of approximate methods, see Syski [S60].) Finally, there may be problems that, by virtue of their special structure, can be solved analytically, either in each instance, or asymptotically in some limiting situation. Such problems give qualitative understanding of the possible phenomena that may occur in our class of systems. (One must be warned, however, by the situation in classical mechanics whereby problems, by virtue of being solvable analytically ("integrable"), are prevented from possessing certain traits (such as "chaotic behaviour") common to more general problems.) Analytically solvable problems may also be consulted for quantitative information about similar systems (though usually without any assurance as to the degree of approximation involved). These analytically solvable problems are the subject of our survey in the present paper. We shall begin with some examples that have been fundamental to the development of the theory of stochastic service systems in general.

2. CLASSICAL SYSTEMS

Consider a stream of requests of a single type arriving at a system with rate ν. Let the system contain r resources called "servers", each of which is capable of serving a request with mean service time 1. (We may arrange that the mean service time is 1 by appropriate choice of the unit of time.) Since each of the servers is capable of serving any request, it is immaterial which is chosen when more than one is idle.

A state of the system corresponds to a set of busy servers, but states with the same number of busy servers are equivalent, so instead of analysing a system with 2^r states, we can analyse one with just $r+1$ states. Let us call these states $S_0, \ldots, S_r$, where j servers are busy in S_j. If we are in state S_j, and $j < r$, then we make a transition to state S_{j+1} at rate ν (the rate at which a new request arrives to engage another server). If we are in state S_j, and $j > 0$, then we make a transition to state S_{j-1} at rate j (the rate at which one of j requests completes service and releases a server).

Since the state transition diagram is finite and strongly connected, the process is ergodic and there is a unique equilibrium distribution, which we can discover in the following way. Let P_j denote the probability of being in state S_j. Transitions from the set $\{S_0, \ldots, S_{j-1}\}$ to the set $\{S_j, \ldots, S_r\}$ alternate with transitions in the reverse direction, so in equilibrium both types of transition must occur at equal rates. Transitions from $\{S_0, \ldots, S_{j-1}\}$ to $\{S_j, \ldots, S_r\}$ are transitions from S_{j-1} to S_j, however, and similarly in the reverse direction. Thus we must have

$$\nu P_{j-1} = j P_j \quad \text{for } 1 \leq j \leq r.$$

This gives us r equations among $r+1$ unknowns, and the normalisation condition $\sum_{0 \leq j \leq r} P_j = 1$ completes the specification of the unique equilibrium distribution.

These equations are easily solved to give

$$P_j = \frac{\frac{\nu^j}{j!}}{\sum_{0\le i\le r} \frac{\nu^i}{i!}}.$$

This distribution is called the Erlang distribution [Er18]. It is a truncated Poisson distribution, which emerges upon taking $r = \infty$, whence the sum in the denominator becomes e^ν.

The rate at which requests are lost is equal to the probability P_r of being in the state S_r in which requests are lost, times the rate ν at which requests arrive when the system is in this state. Thus the fraction of the requests that are lost is simply

$$B(r,\nu) = \frac{\frac{\nu^r}{r!}}{\sum_{0\le i\le r} \frac{\nu^i}{i!}}.$$

There is a remarkable identity expressing this fraction in terms of an integral rather than a sum, $B(r,\nu) = \nu^r e^{-\nu} / \int_\nu^\infty x^r e^{-x}\,dx$, which we commend to the reader as an exercise. (The integral in the denominator is an "incomplete gamma function".) This representation has the merit of showing that $B(r,\nu)$ is an analytic function of r as well as ν, a property that will prove useful later.

A fundamental property of our class of problems is that the arrival rate for requests is independent of the state of the system. It often happens, however, that the arrival rate is diminished in proportion to the number of requests being served (as in a telephone exchange where subscribers, who cannot put calls on "hold", do not generate new requests while they are engaged in a call). There is a simple device, however, whereby this effect can be modelled by problems in our class.

Consider n streams of requests of types $R_1, \ldots, R_n$, each arriving at a system with rate λ. Let the system contain n resources $C_1, \ldots, C_n$ called "customers", as well as r servers. A request of type R_m can be served only by a combination comprising the specific customer C_m and one (it is immaterial which) of the r servers. Again the mean service time is 1.

The states of the system again correspond to the numbers of busy servers, but now if we are in state S_j, we make a transition to state S_{j+1} at rate $(n-j)\lambda$ (the number of idle customers, times the arrival rate for requests at each idle customer). Proceeding as before we obtain the equilibrium distribution $P_j = \binom{n}{j}\lambda^j / \sum_{0\le i\le r}\binom{n}{i}\lambda^i$. This distribution is called the Engset distribution [En18]. If we set $p = \lambda/(1+\lambda)$, it can be rewritten as

$$P_j = \frac{\binom{n}{j}p^j(1-p)^{n-j}}{\sum_{0\le i\le r}\binom{n}{i}p^i(1-p)^{n-i}}.$$

This reveals it as a truncated Bernoulli distribution, which emerges upon taking $r = n$, whence the sum in the denominator becomes 1. We can also regard the Erlang distribution as a limiting case of the Engset distribution in which $n \to \infty$ and $\lambda \to 0$ in such a way that $n\lambda = \nu$.

Let us say that a request is "active" if when it arrives, its customer is idle. Since we are interested in the effect of the limited number of servers ($r < n$), we seek the fraction of active requests that are lost, rather than the fraction of all requests. To determine the fraction $A(r,n,\lambda)$ of active requests that are lost, we divide the rate at which active requests are lost, $(n-r)\lambda P_r$, by the rate at which active requests arrive, $\sum_{0\leq j\leq r}(n-j)P_j$. This gives

$$A(r,n,\lambda) = \frac{\binom{n-1}{r}p^r(1-p)^{n-1-r}}{\sum_{0\leq j\leq r}\binom{n-1}{j}p^j(1-p)^{n-1-j}}.$$

(There is again an integral representation, involving the "incomplete beta function".) The Erlang and Engset distributions play an important role not only in the exact analysis of simple systems, but also in the approximate analysis of more complex systems such as "link systems" (see Jacobæus [J50]).

Consider the lost requests from an Erlang system (with parameters r and ν) or the lost active requests from an Engset system (with parameters r, n and λ). Suppose that these requests are presented to a second, "overflow", system that has m servers. Here m may be large enough to serve all requests that arrive ($m=\infty$ for the Erlang system, $m=n-r$ for the Engset), or it may be smaller, so that requests are lost from the overflow system as well as from the "primary" system. What is the distribution of the number of requests being serviced by the overflow system?

To analyse this situation we must consider the two systems as a whole, and the states of this combined system (after grouping equivalent states) form a two-dimensional array:

$$\begin{matrix} S_{0,0} & S_{0,1} & \cdots & S_{0,m} \\ S_{1,0} & S_{1,1} & \cdots & S_{1,m} \\ \vdots & \vdots & \ddots & \vdots \\ S_{r,0} & S_{r,1} & \cdots & S_{r,m} \end{matrix}$$

(there are j primary servers and k overflow servers busy in state $S_{j,k}$). Transitions are possible only from a state to one of its four neighbours at the cardinal compass points.

Finding the equilibrium distribution for this system is much more difficult than before, but Kosten [Kos37] managed to give expressions for $P_{j,k}$ in the case of an Erlang primary system followed by an infinite overflow system (see also Coffman, Kadota and Shepp [CoKS85]), and Brockmeyer [Br54] extended this to the case of a finite overflow system. The formulæ for the moments of the number of busy servers in the overflow system, together with the analytic continuation of $B(r,\nu)$ to non-integral r form the basis for a powerful method of approximate analysis (the "equivalent random method") due to Wilkinson [W56]. These Erlang overflow systems represent the outermost boundary of exactly solvable overflow systems at the present time, and the corresponding Engset overflow systems seem to lie just beyond this boundary.

3. STATISTICAL MECHANICS

The examples of the preceding section set the stage for what has been the most fruitful source of exactly solvable problems in telephone switching theory: the analogy with statistical mechanics. This analogy was drawn in Beneš's paper 'A "Thermodynamic" Theory of Traffic in Connecting Networks' [Be63].

The analogy is based on the observation that in statistical mechanics one often encounters Markov processes with enormous numbers of states, but one can nevertheless obtain much useful information about what happens in the "thermodynamic limit". Roughly speaking, the thermodynamic limit is one where certain "extensive" parameters (number of particles, volume, internal energy and entropy) tend to infinity in such a way that ratios among them tend to definite limits (particles per unit volume, entropy per particle), while other "intensive" parameters (pressure, temperature) are held constant. One can easily imagine analogous limiting situations in telephone switching theory. For example, in an Erlang system we may let the arrival rate ν and the number of servers r tend to infinity in such a way that their ratio tends to a definite limit μ, the "mean number of requests per server". One can verify directly from the formulæ for $B(r, \nu)$ that in this limit we have $B(r, \nu) \to 0$ if $\mu \leq 1$, and $B(r, \nu) \to 1 - 1/\mu$ if $\mu \geq 1$. (There is a sort of "saturation" effect at $\mu = 1$.)

The methods of statistical mechanics rely heavily on the property of "reversibility", which all the systems it studies possess, but which may or may not be present in a telephone switching problem. Roughly speaking, a system is reversible if it satisfies the following criterion: if one makes a movie of the evolution of a sample of the random process in equilibrium, one cannot determine by statistical tests whether the movie is being run forwards or backwards. In terms of the Markov process describing the system, this amounts to the following: the rate at which transitions from a state S to a state S' occur is equal to the rate at which transitions from S' to S occur. (In the statistical mechanics literature, this is called the "principle of detailed balance".) It is easy to see that this criterion is fulfilled for the Erlang system; indeed, it was by exploiting it that the equations were solved by inspection. On the other hand, it is not fulfilled by the overflow systems considered by Kosten and Brockmeyer: requests can commence service in the overflow system only when all servers in the primary system are busy, but they can complete service at any time, without regard to the state of the primary system. Thus reversibility is a special property, possessed by some but not all telephone switching systems. Its presence, however, greatly simplifies the solution; see Kelly [Kell79] for a general account of the role of reversibility in the theory of stochastic service systems.

Kolmogorov [Kol36] gave the following necessary and sufficient condition for a Markov process to be reversible. For each pair of states (S, S'), define the ratio $\varrho(S, S') = \sigma(S, S')/\sigma(S', S)$, where $\sigma(S, S')$ is the rate at which the system makes a transition into state S' when it is in state S. (We have $\varrho(S, S') = \varrho(S', S)^{-1}$ and $\varrho(S, S) = 1$.) Then the process is reversible if and only if, for every cycle $S_1, S_2, \ldots, S_n, S_1$, we have $\varrho(S_1, S_2)\, \varrho(S_2, S_3) \cdots \varrho(S_n, S_1) = 1$. If this criterion is fulfilled, then we may define a function τ on the states by choosing a reference state S_1, setting $\tau(S_1) = 1$, and then, for every other state S_n, setting $\tau(S_n) = \varrho(S_1, S_2)\, \varrho(S_2, S_3) \cdots \varrho(S_{n-1}, S_n)$ for some path $S_1, S_2, \ldots, S_n$ from S_1 to S_n. (Kolmogorov's criterion ensures that this definition does not depend on the choice of path.) The equilibrium probability P_j of being in state S_j is then given by normalising the function τ: $P_j = \tau(S_j)/\sum_i \tau(S_i)$. (The normalisation ensures that this distribution does not depend on the choice of reference state.) The sum in the denominator is called the "partition function" for the system; it is usually a function of one or more parameters such as the temperature or arrival rate, though we have not indicated this dependence explicitly.

It statistical mechanics, it is common to write $\tau(S_j) = e^{-H(S_j)/T}$, where $H(S_j)$ is the "energy level" of state S_j and T is the temperature (in units for which "Boltzmann's constant" is 1). In telephone switching, it is more common to write $\tau(S_j) = \lambda^{H(S_j)}$, where λ is the arrival rate and $H(S_j)$ is the number of requests in service (the "occupancy") in state S_j. In either case P_j is called the "Gibbs distribution" associated with the potential function $H(S_j)$.

Although it is possible to define $T = -1/\log\lambda$ as the "temperature" of a telephone switching system, this leads to some unintuitive consequences: there are often negative temperatures, which turn out to be "hotter" than positive temperatures. It is better to adopt a "chemo-mechanical" rather than "thermo-mechanical" nomenclature, in which λ is taken as the fundamental independent variable. (Physicists refer to λ as the "activity", or to its reciprocal as the "fugacity".)

Beneš [Be63] acknowledged three drawbacks to the analogy with statistical mechanics. Two of these, the assumption of arrival rates that may be unrealistic, and the ability to deal with at most one of the possible policies for assigning resources to requests, stem directly from the requirement of reversibility. The third is present despite, rather than because of reversibility: "The problem of calculating the partition function [...] is, as in statistical mechanics, very difficult except in cases of unrealistic simplicity." In the remainder of this section we shall survey the cases in which this drawback has been surmounted and the calculation of the partition function has been carried out far enough to yield interesting asymptotic information.

If we seek situations in which we might determine a thermodynamic limit, two main classes emerge. In one, the number of types of requests and resources remains fixed, while the rate at which each type of request arrives, and the number of resources of each type, tend to infinity in such a way that the ratios among them tend to definite limits. The thermodynamic limit of the Erlang system, described above, is of this type. A broad generalisation of this situation has been described by Kelly [Kell86]. Suppose that each type of request requires for service some fixed number of resources of each type, but that (beyond the equivalence of resources of the same type) no other alternatives exist. (Kelly calls this condition "fixed routing".) Then in the limiting situation described above, the fraction of the requests of each type that are lost tends to a definite limit, and these limiting values are the unique solution of a system of equations that can be written down by inspection from the data of the problem. This result permits the asymptotic analysis of many interesting systems, but the restriction to fixed routing is a strong one. It is plausible that an analogous result exists for a broader condition called "hierarchical routing", in which the types of resources are assigned to "levels", each type of request can be satisfied by any of a sequence of combinations of resources, all of the resources in a given combination are at the same level, and combinations of lower level receive preference over combinations of higher level. A still more general condition called "non-hierarchical routing" appears to lead to thermodynamic limits that violate the uniqueness present with fixed routing (see Nakagome and Mori [NM73] and Marbukh [Marb81]), but such systems have thus far resisted rigorous analysis.

In the second class of thermodynamic limits, the number of types of requests and the number of types of resources tend to infinity, while the arrival rate for each type of request and the number of resources of each type are held constant. This class corresponds more closely than the first to traditional statistical mechanics, and two large subclasses (which physicists call "monomer" systems and "dimer" systems) have drawn attention from the communication community. A monomer system is based on an undirected graph (finite or infinite), with types of requests corresponding to vertices, and types of resources corresponding to edges. In the simplest case, there is just one resource of each type, and the arrival rate λ is the same for each type of request. A request requires for service all the resources that are incident with it (no alternatives are possible). A dimer system is similar, but the resources correspond to vertices and the requests correspond to edges. (There is an intimate connexion between Gibbs distributions for systems on graphs such as these and what are called "Markov random fields"; see Preston [Pr74] for an introduction to this vast literature and Kelbert and Sukhov [KelbS83] for an application to communication networks.)

We may consider an infinite graph G to be the limit of a sequence of finite graphs G_n, which are usually subgraphs or quotient graphs of G. Let $\Psi_n(\lambda)$ denote the partition function for the system on G_n. If $\Psi_n(\lambda)^{1/n}$ tends to a definite limit $\psi(\lambda)$ as $n \to \infty$, we say that the thermodynamic limit exists. The limiting value $\psi(\lambda)$ is called the "partition function per unit"; equivalently $\log \psi(\lambda)$ is called the "free energy per unit". (Usually n denotes the number of vertices or edges in G_n, in which case we may speak of the partition function or free energy per vertex or per edge.)

In chemo-mechanical terminology, the free energy per unit may be identified with the "pressure" for many systems. The occupancy per unit (which physicists call the "density") is then given by $p = \lambda\partial(\log \psi(\lambda))/\partial\lambda$. The variance per unit is given by the next derivative, $\sigma^2 = \lambda\partial\lambda\partial(\log \psi(\lambda))/\partial\lambda^2$. (Physicists usually refer to the "compressibility" σ^2/p^2, which is equal to the derivative of density with respect to pressure, rather than to the variance.) Finally, the entropy per unit is $\eta = \log \psi(\lambda) - p \log \lambda$. (The Gibbs distribution is that distribution which maximises the entropy $-\sum_j P_j \log P_j$ among those with a given expected occupancy $\sum_j H(S_j)P_j$.)

As a simple example, consider the path Π_n in which n vertices alternate with $n-1$ edges. The monomer system on Π_n is isomorphic to the dimer system on Π_{n+1}; for definiteness, we shall consider the monomer system. A state of the system corresponds to a set of "occupied" vertices, no two of which are adjacent. It is easy to see that the number of states with k occupied vertices is $\binom{n-k}{k}$, so that the partition function is $\Psi_n(\lambda) = \sum_{k\geq 0} \binom{n-k}{k}\lambda^k$. (This partition function can be expressed in terms of the Chebyshev polynomials of the second kind.) Noting that $\Psi_0(\lambda) = \Psi_1(\lambda) = 1$ and $\Psi_{n+2}(\lambda) = \Psi_{n+1}(\lambda) + \lambda\Psi_n(\lambda)$, we see that $\Psi_n(\lambda)$ is the coefficient of z^n in the generating function $\Phi(\lambda, z) = (1+2z)/(1-z-\lambda z^2)$ (which physicists call the "grand partition function"). Other systems that can similarly be solved for their partition function include (1) the monomer or dimer system on the cycle C_n, in which n vertices alternate with n edges (this involves the Chebyshev poynomials of the first kind), (2) the dimer system for a balanced tree (this also involves the Chebyshev polynomials of the second kind), (3) the dimer system for the complete graph, with n vertices and $\binom{n}{2}$ edges (this involves the Hermite polynomials) and (4) the complete bipartite graph with $n+m$ vertices and nm edges (this involves the generalised Laguerre polynomials). See Heilmann and Lieb [HeL72], Godsil and Gutman [GoG81] and Pinsky and Yemini [PinY84].

For a problem that is solvable in the sense of the preceding paragraph (where a grand partition function can be written down), it is straightforward to prove the existence of the thermodynamic limit and to determine the limiting value of the partition function per server. The procedure is to use Cauchy's contour integral to extract the partition function from the grand partition function, then to use the saddle-point method to determine the asymptotics of the integral. For the path, the partition function per vertex turns out to be $\psi(\lambda) = (1+\sqrt{1+4\lambda})/2$, and the occupancy per vertex is given by $p = 1/2 - 1/2\sqrt{1+4\lambda}$. The path and the cycle are typical of a large class of "one-dimensional" systems that can be solved exactly in this way. Furthermore, both the path and the cycle may be regarded as tending to a common infinite limit graph as $n \to \infty$: the "infinite path" or "infinite cycle", in which vertices correspond to integers and edges correspond to pairs of integers that differ by 1. We may regard the limiting partition function per vertex as referring to this infinite system.

Our final example of an exactly solvable system comes directly from the statistical mechanics literature, with only a change of terminology. It may be described as "cellular radio-telephony with one frequency". Imagine a cellular radio-telephone system in which space, taken to be the Euclidean plane, is tessellated by regular hexagons in the usual "honeycomb" fashion. Suppose that requests to use one available frequency for communication arrive at cells, and that such a request can be served only if, at the time of its arrival, no request is being served in any of the six adjacent cells. We then have a monomer system on the infinite triangular lattice; the vertices correspond to cells, and the edges correspond to the frontiers between cells. (We do not consider the requests to be mobile, or their servers to be subject to rearrangement.) We may now consider a given arrival rate λ at each cell, and ask for the probability p that a cell is occupied. This is an infinite system, but we may consider a sequence of finite subsystems by considering either subgraphs (giving what physicists call "free boundary conditions") or quotient graphs (giving "periodic boundary conditions") of the infinite graph; for suitable sequences, we may prove that the thermodynamic limit exists.

The system just described is isomorphic to what physicists call the "lattice gas of hard hexagons", and it is one of a small number of models for which an exact expression can be written down for the partition function per cell. (To be precise, it should be said that, although there is a variety of evidence affording a considerable certainty that the solution is correct, this correctness has not been proved with mathematical rigour.) The solution was discovered by Baxter [Ba82]; a particularly interesting feature is that there is a critical value $\lambda_0 = (11 + 5\sqrt{5})/2 = 11.09017\ldots$ of the arrival rate per cell λ at which a "phase transition" occurs, with the solution having different analytic expressions above and below this value. (The "saturation" of the Erlang system provides a trivial example of a phase transition, but the present example is much more interesting.)

The nature of the phase transition may be described as follows. If one tries to "pack" requests as tightly as possible into the system, there are three ways to do this, corresponding to the three colours in the minimal colouring of the triangular lattice. When the arrival rate is less than λ_0, there will be small regions that resemble one of these packings, but all three packings will coexist in roughly equal extents. When the arrival rate exceeds λ_0, however, one of the packings will predominate over the other two, and a global order emerges. We may think of the disordered phase as a "gas", and the ordered phase as a "crystal", so the transition corresponds to "crystallisation" or "sublimation".

Heilmann and Lieb [HeL72] have shown that under rather general circumstances, a dimer system cannot have a phase transition. Even for monomer systems that are not exactly solvable, however, it may be possible to prove rigorously the existence of a phase transition. A fundamental technique for this, due to Peierls [Pe36], has been developed greatly in recent years. For examples of its application, see Dobrushin [D68], Heilmann [He72] and Heilmann and Præstgaard [HP74].

4. CONCLUSION

We have considered a rather restricted class of problems in this paper, and we should at least indicate what lies beyond its borders. Allowing refusals would not affect our discussion much; the theorem of Beneš [Be66] would no longer hold, but we would have no new exactly solvable systems to report. Allowing deferrals leads, of course, to the vast domain of queuing theory, which has a disjoint collection of exactly solvable systems of its own. Allowing rearrangements leads to yet another class of problems; for methods that can be brought to bear on dimer systems, see Karp and Sipser [KaS81], and for cellular radio-telephony, see Everitt and Macfadyen [EvM83] and Kelly [Kell86]. Finally, there is a class of problems that as are far from reversible as can be, in which requests arrive and are assigned resources, but never complete service. These are analogous to physical problems of "random sequential adsorption", and the methods used there can be applied to monomer and dimer systems; see Pippenger [Pip89].

We conclude by recapitulating what seem to us the most interesting open problems. Can the overflow from an Engset system be solved in closed form? Under what circumstances does hierarchical routing have a unique thermodynamic limit? What can be said rigorously about non-hierarchical routing? Can the validity of the exact solution for hard hexagons be proved rigorously?

5. REFERENCES

[Ba82] R. J. Baxter, *Exactly Solvable Models in Statistical Mechanics*, Academic Press, London, 1982.

[Be63] V. E. Beneš, 'A "Thermodynamic" Theory of Traffic in Connecting Networks', *Bell Sys. Tech. J.*, 42 (1963) 567–607.

[Be66] V. E. Beneš, "Programming and Control Problems Arising from Optimal Routing in Telephone Networks", *Bell Sys. Tech. J.*, 45 (1966) 1373–1438.

[Br54] E. Brockmeyer, "Det Simple Overflowproblem i Telefontrafikteorien", *Teletechnik*, 5 (1954) 361–374.

[CoKS85] E. G. Coffman, Jr., T. T. Kadota and L. A. Shepp, "A Stochastic Model of Fragmentation in Dynamic Storage Allocation", *SIAM J. Computing*, 14 (1985) 416–425.

[D68] R. L. Dobrushin, "The Problem of the Uniqueness of a Gibbsian Random Field and the Problem of Phase Transitions", *Funct. Anal. Appl.*, 2 (1968) 302–312.

[En18] T. Engset, "Die Wahrscheinlichkeitsrechnung zur Bestimmung der Wahleranzahl in automatischen Fernsprechämtern", *Electrotech. Z.*, 31 (1918) 304–305.

[Er18] A. K. Erlang, "Solution of Some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchnges", *Post Office Elect. Eng. J.*, 10 (1918) 189–197.

[EvM83] D. E. Everitt and N. W. Macfadyen, Analysis of Multicellular Mobile Radiotelephone Systems with Loss", *British Telecomm. Technol. J.*, 1 (1983) 37–45.

[GoG81] C. D. Godsil and I. Gutman, "On the Theory of the Matching Polynomial", *J. Graph Theory*, 5 (1981) 137–144.

[He72] O. J. Heilmann, "Existence of Phase Transitions in Certain Lattice Gases with Repulsive Potential", *Let. al Nuovo Cimento*, 3, 3 (1972) 95–98.

[HeL72] O. J. Heilmann and E. H. Lieb, "Theory of Monomer-Dimer Systems", *Comm. Math. Phys.*, 25 (1972) 190–232.

[HeP74] O. J. Heilmann and E. Præstgaard, 'Phase Transition in a Lattice Gas with Third Nearest Neighbour Exclusion on a Square Lattice", *J. Phys. A: Math., Nucl. Gen.*, 15 (1974) 1913–1917.

[Hw79] F. K. Hwang, "Superior Channel Graphs", *Internat. Teletraffic Congr.*, 9 (1979).

[J50] C. Jacobæus, "A Study on Congestion in Link Systems", *Ericsson Technics*, 48 (1950) 1–68.

[KaS81] R. M. Karp and M. Sipser, "Maximum Matchings in Sparse Random Graphs", *IEEE Symp. on Foundations of Computer Science*, 22 (1981) 364–375.

[KelbS83] M. Ya. Kelbert and Yu. M. Sukhov, "Existence and Uniqueness Conditions for a Random Field Describing the State of a Switching Network", *Problems of Info. Transm.*, 19 (1983) 289–304.

[Kell79] F. P. Kelly, *Reversibility and Stochastic Networks*, Wiley, Chichester, 1979.

[Kell86] F. P. Kelly, "Blocking Probabilities in Large Circuit-Switched Networks", *Adv. Appl. Prob.*, 18 (1986) 473–505.

[Kol36] A. N. Kolmogorov, "Anfangsgründe der Theorie der Markoffschen ketten mit unendlich vielen möglichen Zuständen", *Mat. Sbornik*, 1 (1936) 607–610.

[Kos37] L. Kosten, "Über Sperrungswahrscheinlichkeiten bei Staffelschaltungen", *Elektr. Nachr.-Tech*, 14 (1937) 5–12.

[Marb81] V. V. Marbukh, "Asymptotic Investigation of a Complete Communication Network with a Large Number of Points and Bypass Routes", *Problems of Info. Transm.*, 7 (1981) 89–95.

[NM73] Y. Nakagome and H. Mori, "Flexible Routing in the Golbal Communication Network", *Internat. Teletraffic Congr.*, 7 (1973) 426.

[Pe36] R. Peierls, "Ising's Model of Ferromagnetism", *Proc. Cambridge Phil. Soc.*, 32 (1936) 477–481.

[PinY84] E. Pinsky and Y. Yemini, "A Statistical Mechanics of Some Interconnection Networks", *Performance '84*, 147–158.

[Pip89] N. Pippenger, "Random Sequential Adsorption on Graphs", *SIAM J. Discr. Math.*, 2 (1989) 393–401.

[Pr74] C. Preston, *Gibbs States on Countable Sets*, Cambridge University Press, London, 1974.

[S60] R. Syski, *Introduction to Congestion Theory in Telephone Systems*, Oliver and Boyd, Edinburgh, 1960; second edition: North-Holland, Amsterdam, 1986.

[W56] R. I. Wilkinson, "Theories for Toll Traffic Engineering in the USA", *Bell Sys. Tech. J.*, 35 (1956) 421–514.

ANALYSIS AND SYNTHESIS OF A CLASS OF DISCRETE-TIME NEURAL NETWORKS DESCRIBED ON HYPERCUBES*

A. N. MICHEL, J. SI, G. YEN
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

I. INTRODUCTION

We consider neural networks described by a system of first order linear difference equations which are defined on a closed hypercube. We refer to such systems as " linear systems in a saturated mode", (LSSM). Specificallly, we consider neural networks described by equations of the form,

$$x(k+1)=Tx(k)+I \qquad k=0,1,2,... \tag{S}$$

with the constraints

$$-1\leq x_i\leq 1, \qquad i=1,...,n$$

where $x=(x_1,...,x_n)^T\in D^n=\{x\in R^n: -1\leq x_i\leq 1, i=1,2,...,n\}$, $T=[T_{ij}]$ is an $n\times n$ constant matrix, and $I=(I_1,..., I_n)^T$ is a constant vector. The principal difference between system (S) and the usual systems described by linear difference equations is that the latter are defined on open subsets of R^n while the former is defined on the closed subset D^n of R^n. We will show that an LSSM system may be viewed as a reduced order linear system. In general, the solutions of system (S) will be in a saturated mode for some or for all of the discrete points in time, k=0,1,2,.... System (S) can easily be implemented on a digital computer or in digital hardware.

The results which we establish for system (S) fall into two categories. One type of result addresses analysis while the other type pertains to synthesis procedures for system (S). Due to space limitations, the proofs of our results will be given elsewhere.

II. ANALYSIS

Before giving a proper definition of solution for system (S), we need to introduce some additional concepts.

For each integer m, $0\leq m\leq n$, let

$$\Delta_m= \{\xi=(\xi_1,...,\xi_n)^T\in \Delta : \xi_{\mu(i)}=0,\ 1\leq i\leq m \text{ and } \xi_{\mu(i)}=\pm 1,\ m<i\leq n, \text{ for some } \mu\in \mathrm{Sym}(n)\}, \tag{2.1a}$$

and let

$$\Delta=\{\xi=(\xi_1,...,\xi_n)^T : \xi_i=\pm 1 \text{ or } 0,\ 1\leq i\leq n\}. \tag{2.1b}$$

The number of elements in Δ_m is equal to $[n!/(m!(n-m)!)]2^{n-m}$, and the number of elements in $\Delta=\cup\Delta_m$, $0\leq m\leq n$, is 3^n. For each $\xi\in \Delta$, let

$$C(\xi)=\{x=(x_1,...,x_n)^T\in R^n : |x_i|<1 \text{ if } \xi_i=0, \text{ and } x_i=\xi_i \text{ if } \xi_i\neq 0\}. \tag{2.1c}$$

From the notation given above, we have the following result.

Lemma 2.1. 1) $\Delta_0=B^n$ and $C(\xi)=\{\xi\}$, for any $\xi\in \Delta_0$.

2) $\Delta_n=\{0\}$ and $C(0)=(D^n)^0=\{x\in R^n: -1<x_i<1, i=1,...,n\}$.

3) $\partial(D^n)=\cup C(\xi)$, $\xi\in \cup\Delta_m$, $0\leq m<n$.

4) $D^n=\cup C(\xi)$, $\xi\in \cup\Delta_m$, $0\leq m\leq n$.

5) For any $\xi, \eta\in \Delta$, $\xi\neq\eta$, $C(\xi)\cap C(\eta)=\varnothing$.

Suppose that $\xi\in \Delta_m$ and $\mu\in \mathrm{Sym}(n)$ such that,

$$\xi_{\mu(i)}=0,\ 1\leq i\leq m \text{ and } \xi_{\mu(i)}=\pm 1,\ m<i\leq n. \tag{2.2}$$

Subsequently, we will make use of the notation

---

*The work of A. N. Michel, J. Si and G. Yen is supported in part by the National Science Foundation under grant ECS 88-02924.

$$T_{I,I}=[T_{\mu(i)\mu(j)}], \qquad \text{for } 1\le i,j\le m$$
$$T_{I,II}=[T_{\mu(i)\mu(j)}], \qquad \text{for } 1\le i\le m, \ \ m<j\le n$$
$$T_{II,I}=[T_{\mu(i)\mu(j)}], \qquad \text{for } m<i\le n, \ \ 1\le j\le m$$
$$T_{II,II}=[T_{\mu(i)\mu(j)}], \qquad \text{for } m<i,j\le n$$
$$T=\begin{bmatrix} T_{I,I} & T_{I,II} \\ T_{II,I} & T_{II,II} \end{bmatrix} \tag{2.3a}$$

and

$$I_I=(I_{\mu(1)},\dots,I_{\mu(m)})^T, \qquad I_{II}=(I_{\mu(m+1)},\dots,I_{\mu(n)})^T, \tag{2.3b}$$

and

$$\xi_I=(\xi_{\mu(1)},\dots,\xi_{\mu(m)})^T, \qquad \xi_{II}=(\xi_{\mu(m+1)},\dots,\xi_{\mu(n)})^T. \tag{2.3c}$$

Remark 2.1. 1) For a given $\xi\in\Delta_m$, there may exist different permutations in Sym(n) for which (2.2) is true. For these different permutations, the notation given above will be the same up to different orders in components. Thus, the analysis and conclusions will be identical for any of the permutations used.

2) If m=n, we have $T_{I,I}=T$, $I_I=I$, $\xi_I=\xi$ and $T_{I,II}$, $T_{II,I}$, $T_{II,II}$, I_{II}, ξ_{II} do not exist. If m=0, we have $T_{II,II}=T$, $I_{II}=I$, $\xi_{II}=\xi$ and $T_{I,I}$, $T_{I,II}$, $T_{II,I}$, I_I, ξ_I do not exist.

Definition 2.1. 1) Consider $\xi\in\Delta_m$, $0<m\le n$, with $\mu\in$ Sym(n), such that $\xi_{\mu(i)}=0$, $1\le i\le m$, and $\xi_{\mu(i)}=1$ or -1, $m<i\le n$. Also, consider the linear system defined by

$$x_I(k+1)=T_{I,I}x_I(k)+T_{I,II}\xi_{II}+I_I \tag{S_ξ}$$

where $x_I=(x_{\mu(1)},\dots,x_{\mu(m)})^T$, $x_{II}=(x_{\mu(m+1)},\dots,x_{\mu(n)})^T$ and $-1<x_{\mu(i)}<1$, for $1\le i\le m$. System (S_ξ) is said to be the reduced linear system of system (S) over the region C(ξ).

2) For any $\xi\in\Delta_m$, a function φ: $\{0,1,\dots k_0\}\to C(\xi)$, is said to be a (local) solution of system (S) if the vector function φ_I containing the μ(i)-th components of φ, $1\le i\le m$, is a solution of the linear system (S_ξ), i.e.,

$$\varphi_I(k+1)=T_{I,I}\varphi_I(k)+T_{I,II}\xi_{II}+I_I, \qquad k\in\{0,1,\dots,k_0\} \tag{2.4a}$$

and

$$\min((T_{II,I}\varphi_I(k)+T_{II,II}\xi_{II}+I_{II})*\xi_{II})\ge 1, \qquad k\in\{0,1,\dots,k_0\} \tag{2.4b}$$

where

$\varphi_I=(\varphi_{\mu(1)},\dots,\varphi_{\mu(m)})^T$ and

$\varphi_{II}=(\varphi_{\mu(m+1)},\dots,\varphi_{\mu(n)})^T$.

and where $x*y=(x_1y_1,\dots,x_ny_n)$, for $x,y\in R^n$.

In particular, if $\xi\in\Delta_m$, m<n, the solution φ is said to be in a saturated mode.

Remark2.2. 1) $\varphi(\{0,1,\dots,k_0\})\subset C(\xi)$ implies that

$-1<\varphi_{\mu(i)}(k)<1$, $i=1,\dots,m$, $k\in\{0,1,\dots,k_0\}$,

$\varphi_{\mu(i)}(k)=\xi_{\mu(i)}$, $i=m+1,\dots,n$, $k\in\{0,1,\dots,k_0\}$, i.e., $\varphi_{II}(k)=\xi_{II}$ when $k\in\{0,1,\dots,k_0\}$.

2) When m=n, $\Delta_n=\{0\}$. In this case, (3.4b) has no meaning and we only need to consider (2.4a). Furthermore, the (local) solutions in $C(0)=(D^n)^0=\{x\in R^n: -1<x_i<1, i=1,\dots,n\}$ defined above for the LSSM system (S) will be identical to the usual solutions defined on the open subset C(0) for the linear system (S).

3) When m=0, $\Delta_0=B^n$ and for $\xi\in\Delta_0$, $C(\xi)=\{\xi\}$. In this case, (2.4a) does not exist and we only need to consider (2.4b). Furthermore, the only function with the range in C(ξ) is the constant function $\varphi(k)\equiv\xi$.

Definition 2.2. For $x_0\in D^n$, a funtion $\varphi=\varphi(.,x_0)$: $Z^+\to D^n$ is said to be a solution of (S) starting at x_0 if

a) $\varphi(0,x_0)=x_0$, and

b) there are countably many sets $E_i=\{I_i,\ldots,I_i+k_i\}$, $I_i\in Z^+$, $k_i\in Z^+$, $I_0=0$ such that $E_i\cap E_j=\varnothing$ when $i\neq j$, and $\cup E_i=Z^+$, and $\varphi(k)$ restricted to each E_i is a (local) solution as defined in Definition 2.1.

Definition 2.3. 1) A vector $x_e\in D^n$ is said to be an equilibrium of system (S) if the function $\varphi=\varphi(.,x_e)$: $Z^+\rightarrow D^n$ defined by $\varphi(k,x_e)\equiv x_e$ is a solution of (S).

2) Let x_e be an equilibrium point of system (S),

i) x_e is said to be stable if for every $\varepsilon > 0$, there is a $\delta=\delta(\varepsilon) > 0$ such that $|\varphi(k,x)-x_e| < \varepsilon$, for all $k\in Z^+$ whenever $|x-x_e| < \delta$

ii) x_e is said to be asymptotically stable if it is stable and if there is an $\eta > 0$ such that $\lim_{k\rightarrow\infty}|\varphi(k,x)-x_e| = 0$ whenever $|x-x_e| < \eta$.

iii) x_e is said to be unstable if it is not stable.

We will have occasion to make use of the following assumptions for system (S).

Assumption 2.1. 1) T is symmetric.

2) For any m, $0\leq m\leq n$, and for any $\xi\in\Delta_m$, the $m\times m$ matrix $(E_m-T_{I,I})=(E_m-T_{\mu(i)\mu(j)})$, $1\leq i,j\leq m$ is non-singular, where E_m is the $m\times m$ identity matrix, and $\mu\in \mathrm{Sym}(n)$, such that $\xi_{\mu(i)}=0$, $1\leq i\leq m$, and $\xi_{\mu(i)}=\pm1$, $m<i\leq n$.

For system (S) satisfying Assumption 2.1 , we introduce the following notation.

I. For $\xi=0\in\Delta_n$, let

$$x_\xi=Tx_\xi+I\in R^n \text{ or } x_\xi=(E_n-T)^{-1}I \tag{2.5a}$$

where E_n denotes the $n\times n$ identity matrix.

II. For $\xi\in\Delta_m$, $0<m<n$ and with $T_{I,I},\ldots,T_{II,II}$, I_I,I_{II} defined as in equation (2.4), let

$$x_\xi=(x_{\xi1},\ldots,x_{\xi n})^T\in R^n \tag{2.5b}$$

where

$$\begin{aligned} x_{\xi I}&=(x_{\xi\mu(1)},\ldots,x_{\xi\mu(m)})^T\\ &=(E_m-T_{I,I})^{-1}(T_{I,II}\xi_{II}+I_I)\\ x_{\xi II}&=(x_{\xi\mu(m+1)},\ldots,x_{\xi\mu(n)})^T=\xi_{II}\end{aligned}$$

and let

$$r_\xi\in R,\ r_\xi=\min((T_{II,I}x_{\xi I}+T_{II,II}\xi_{II}+I_{II})*\xi_{II}). \tag{2.5c}$$

III. For $\xi\in\Delta_0=B^n$, $C(\xi)=\{\xi\}$, let

$$x_\xi=\xi \tag{2.5d}$$

and

$$r_\xi\in R,\ r_\xi=\min((T\xi+I)_*\xi) \tag{2.5e}$$

Assumption 2.2. Given Assumption 2.1, assume that for $\xi\in\Delta_m$, $0\leq m<n$, $x_\xi\notin\partial C(\xi)$, i.e., $x_{\xi\mu(i)}\neq\pm1$, for $i=1,\ldots,m$.

Remark 2.3. For fixed T, Assumption 2.2 is true for almost all vectors I in R^n.

Lemma 2.2. Suppose that system (S) satisfies Assumptions 2.1 and 2.2. Then for any $\xi\in\Delta_m$, $0\leq m<n$, $r_\xi\neq1$.

Theorem 2.1. suppose that system (S) satisfies Assumptions 2.1 and 2.2. Then for any m, $0\leq m\leq n$, and for any $\xi\in\Delta_m$, with the notation given above, we have the following results:

Case I. $m=n$, $\xi=0\in\Delta_n$.

1) If $x_\xi\notin C(\xi)=(D^n)^0$, there is no equilibrium point of system (S) in $(D^n)^0$.

2) If $x_\xi\in C(\xi)=(D^n)^0$, x_ξ is the unique eqilibrium point of system (S) in $(D^n)^0$. In particular,

i) If the eigenvalues of $T=T_{I,I}$ satisfy $|\lambda_i|<1$, i=1,...,n, then x_ξ is asymptotically stable and there are no other equlibrium points of system (S) in $\overline{C(\xi)}=D^n$; and

ii) If there is at least one i such that $|\lambda_i|>1$, then x_ξ is unstable.

Case II. $0<m<n$, $\xi\in\Delta_m$.

A) If $r_\xi<1$, there is no equilibrium point of system (S) in $C(\xi)$.

B) If $r_\xi>1$, we have that

1) If $x_\xi\notin C(\xi)$, there is no equilibrium point of system (S) in $C(\xi)$.

2) If $x_\xi\in C(\xi)$, x_ξ is the unique equilibrium point of system (S) in $C(\xi)$. In particular,

i) If the eigenvalues of $T_{I,I}$ satisfy $|\lambda_i|<1$, i=1,...,m, then x_ξ is asymptotically stable, and there are no other equilibrium points in $\overline{C(\xi)}$; and

ii) If there is at least one i such that $|\lambda_i|>1$, then x_ξ is unstable.

Case III. $\xi\in\Delta_0=B^n$. (Note: $C(\xi)=\{\xi\}$)

A) If $r_\xi<1$, $x_\xi=\xi$ is not an equilibruim point of system (S).

B) If $r_\xi>1$, $x_\xi=\xi$ is an asymptotically stable equilibrium point of system (S).

Remark 2.4. 1) Theorem 2.1 establishes an algorithm to locate all equilibrium points of (S) and to determine the stability properties of all equilibrium points.

2) In practice, due to noise, unstable equilibrium points can not be used as memory locations. However, by investigating the locations of the unstable equilibrium points of (S), the domain of attraction of each asymptotically stable equilibrium point of (S) can be approximately determined.

Theorem 2.2. If system (S) satisfies Assumption 2.1 and 2.2, we have the following results:

1) There is at most one equilibrium point of (S) in each region $C(\xi)$, $\xi\in\Delta$.

2)There are at most 3^n equilibrium points for system (S).

3) If for $\xi\in\Delta$ the corresponding x_ξ is an asymptotically stable equilibrium point for system (S), then

i) x_ξ is the unique equilibrium point of system (S) in $\overline{C(\xi)}$, and

ii) x_ξ is the unique asymptotically stable equilibrium point of system (S) in $\overline{C(\xi)} \cup C(\zeta)$, where $\zeta\in\Delta$ such that $x_\xi\in\overline{C(\zeta)}$. In particular, if $\xi\in\Delta_0=B^n$ and $x_\xi=\xi$ is an asymptotically stable equilibrium point of system (S), then $x_\xi=\xi$ is the unique asymptotically stable equilibrium point of system (S) in the region $\{x\in D^n: -1<x_i\xi_i\leq 1, 1\leq i\leq n\}$.

4) There are at most 2^n asymptotically stable equilibrium points for system (S).

Remark 2.5. Theorem 2.2 is not only very useful to simplify the algorithm given in Theorem 2.1 for checking the location of equilibrium points of system (S), but it is also useful to determine whether a given set of vectors can be synthesized as a set of asymptotically stable equilibrium points of system (S) or not. (cf. Synthesis Procedure)

Definition 2.4. If T is symmetric, then

1) the energy function E: $D^n\rightarrow R$ of system (S) is defined by

$$E(x)= -x^T(T-E_n)x-2x^TI, \text{ where } E_n \text{ is the identity matrix} \qquad (2.6)$$

and

2) $x\in D^n$ is said to be a (local) minimum of the energy function E if there is an open neighborhood U of x in R^n such that $E(x)\leq E(y)$, for any $y\in D^n\cap U$.

Theorem 2.3. Suppose that system (S) satisfies Assumptions 2.1 and 2.2, and all the eigenvalues of system (S) satisfy $\lambda_i > -1$, i=1,...,n. Then it is true that

1) along non-equilibrium solutions of system (S), the energy function E given in (2.6) decreases monotonically; and

2) each non-equilibrium solution of (S) converges to an equilibrium point of (S).

Remark 2.6. 1) Theorem 2.3 shows that system (S) has no oscillatory solutions.

2) In practice, due to noise, all solutions of system (S) will only converge to asymptotically stable equilibrium points of system (S).

Theorem 2.4. Suppose that system (S) satisfies all assumptions of Theorems 2.1 and 2.3. Then there is a one-to-one correspondence between the set of local minima of the energy function E and the set of asymptotically stable equilibrium points of system (S).

Remark 2.7. Without $\lambda_i>-1$, $i=1,\dots,n$, we can not guarantee the global stability of system (S). As a counter example, let's consider the following system,

$$x_1(k+1)=-2x_1(k)+x_2(k)+I_1$$
$$x_2(k+1)=x_1(k)-2x_2(k)+I_2$$

i.e., $T=\begin{bmatrix}-2 & 1\\ 1 & -2\end{bmatrix}$, $I=\begin{bmatrix}0.1\\ 0.1\end{bmatrix}$. The eigenvalues of T are -1 and -3.

Using equations (2.5), we can find x_ξ's as,

I. m=0, $x_\xi=\begin{bmatrix}0.05\\ 0.05\end{bmatrix}$.

II. m=1, $x_{\xi 1}=\begin{bmatrix}0.367\\ 1\end{bmatrix}$, $x_{\xi 2}=\begin{bmatrix}-0.3\\ -1\end{bmatrix}$, $x_{\xi 3}=\begin{bmatrix}1\\ 0.367\end{bmatrix}$, $x_{\xi 4}=\begin{bmatrix}-1\\ -0.3\end{bmatrix}$.

III. m=2, $x_{\xi 1}=\begin{bmatrix}1\\ 1\end{bmatrix}$, $x_{\xi 2}=\begin{bmatrix}-1\\ -1\end{bmatrix}$, $x_{\xi 3}=\begin{bmatrix}1\\ -1\end{bmatrix}$, $x_{\xi 4}=\begin{bmatrix}-1\\ 1\end{bmatrix}$.

By Theorem 2.1,

Case I, when m=0, $x_\xi=\begin{bmatrix}0.05\\ 0.05\end{bmatrix}$ is not an asymptotically stable equilibrium point.

Case II, when m=1, $x_{\xi 1}=\begin{bmatrix}0.367\\ 1\end{bmatrix}$, we have $r_{\xi 1}=-1.533<1$, and $x_{\xi 1}$ is not asymptotically stable. For $x_{\xi 2}=\begin{bmatrix}-0.3\\ -1\end{bmatrix}$, $r_{\xi 2}=-1.533<1$; $x_{\xi 3}=\begin{bmatrix}1\\ 0.367\end{bmatrix}$, $r_{\xi 3}=-1.533<1$; and $x_{\xi 4}=\begin{bmatrix}-1\\ -0.3\end{bmatrix}$, $r_{\xi 4}=-1.8<1$. Thus $x_{\xi 2}$, $x_{\xi 3}$ and $x_{\xi 4}$ are not asymptotically stable either.

Case III, when m=2, $x_{\xi 1}=\begin{bmatrix}1\\ 1\end{bmatrix}$, $x_{\xi 2}=\begin{bmatrix}-1\\ -1\end{bmatrix}$, $x_{\xi 3}=\begin{bmatrix}1\\ -1\end{bmatrix}$, $x_{\xi 4}=\begin{bmatrix}-1\\ 1\end{bmatrix}$. Corespondingly, $r_{\xi 1}=-0.9<1$, $r_{\xi 2}=-1.1<1$, $r_{\xi 3}=-4.1<1$, $r_{\xi 4}=-3.1<1$. Thus $x_{\xi 1}$, $x_{\xi 2}$, $x_{\xi 3}$ and $x_{\xi 4}$ are not asymptotically stable equilibrium points.

In simulations of this system, none of the initial conditions within the unit square which we tyied converged to any point.

III. SYNTHESIS

In the present section, we address a synthesis procedure for system (S) which is in the spirit of some of the work in [7],[8].

Synthesis problem: Given m vectors in B^n, say $\alpha_1,\dots,\alpha_m$, how can we properly choose a pair {T,I} such that the resulting synthesized system (S) has the properties enumerated below?

1) $\alpha_1,\dots,\alpha_m$ are asymptotically stable equilibrium points of system (S).

2) The system has no oscillatory solutions.

3)The total number of the spurious asymptotically stable equilibrium points (i.e., asymptotically stable equilibrium points of (S) contained in $B^n-\{\alpha_1,\dots,\alpha_m\}$) is as small as possible.

4) The domain of attraction of each α_i is as large as possible.

In view of the analysis results developed in the previous section, we may approach the preceding problem in the following manner.

Synthesis strategy: Given m vectors in B^n, say $\alpha_1,\dots,\alpha_m$, find an n×n matrix $T=[T_{ij}]$ and a vector $I=(I_1,\dots,I_n)^T$ such that:

1) T satisfies Assumption 3.1.

2) $x_{\alpha i}=\alpha_i$, where $\tau_1 x_{\alpha i}=T\alpha_i+I$, and $\tau_1>1$. (3.1)

3) T has repeated eigenvalues equal to τ_1 and $-\tau_2$, $|\tau_2|<1$, and τ_2 should be as large as possible.

Remark 3.1. In the following, we give the rational for the above synthesis strategy. With Assumption 2.1 and by Remark 2.3, we may assume that Assumption 2.2 is also true for the synthesized system (S). Then all of the results in the previous section are applicable. We have:

1) By Theorem 2.3, the synthesized system (S) will have no oscillatory solutions.

2) By Theorem 2.1, for $1\leq i\leq m$, $x_{\alpha i}=\alpha_i$ is an asymptotically stable equilibrium point of the synthesized system (S).

3) As shown in Lemma 3.2 below, the larger τ_2 is ($-1<\tau_2<1$), the fewer spurious asymptotically stable equilibrium points for system (S) will exist. In general, this will result in larger domains of attraction for the remaining desired asymptotically stable equilibrium points.

Suppose that we are given m vectors in B^n, say $\alpha_1, \ldots, \alpha_m$. Let

$$L=\mathrm{Span}(\alpha_1-\alpha_m,\ldots,\alpha_{m-1}-\alpha_m) \tag{3.2a}$$

and

$$L_a=\mathrm{Aspan}(\alpha_1,\ldots,\alpha_m). \tag{3.2b}$$

Then L is the linear subspace of R^n generated by the (m-1) vectors $\alpha_1-\alpha_m,\ldots,\alpha_{m-1}-\alpha_m$, L_a is the affine subspace of R^n generated by the vectors $\alpha_1,\ldots,\alpha_m$, and $L_a=L+\alpha_m$. Assume that $k=\mathrm{rank}(L)$, $\{u_1,\ldots,u_k\}$ is an orthonormal basis of L, and $\{u_{k+1},\ldots,u_n\}$ is an orthonormal basis of $L^\perp$. Let $U^+=[u_1,\ldots,u_k]$, $U^-=[u_{k+1},\ldots,u_n]$,

$$T^+=[T_{ij}{}^+]=\Sigma_{i=1}^{k}\, u_i u_i{}^T=U^+(U^+)^T, \tag{3.3a}$$

$$T^-=[T_{ij}{}^-]=\Sigma_{i=k+1}^{n}\, u_i u_i{}^T=U^-(U^-)^T, \tag{3.3b}$$

and let

$$T_\tau=\tau_1 T^+-\tau_2 T^-, \text{ and } I_\tau=\tau_1\alpha_m-T_\tau\alpha_m, \tag{3.4}$$

where $\tau_1,\tau_2\in R$ are parameters. For this class of synthesized LSSM systems

$$x(k+1)=T_\tau x(k)+I_\tau, \qquad k=0,1,2,\ldots \tag{S_τ}$$

with the constraints

$-1\leq x_i\leq 1$, $i=1,\ldots,n$,

we have the following result.

Lemma 3.1. 1) T^+ and T^- depend only on the given set $\{\alpha_1,\ldots,\alpha_m\}$ and they are independent of the choice of α_m and of the orthonormal basis $\{u_1,\ldots,u_n\}$ of R^n. This is also true for T_τ and I_τ, for any $\tau_1,\tau_2\in R$.

2) For any τ_1 and τ_2, T_τ is symmetric.

3) For any τ_1, τ_2 and for any $\alpha\in L_a$, $T_\tau\alpha+I_\tau=\tau_1\alpha$. In particular, for each α_i, $i=1,\ldots,m$, $T_\tau\alpha_i+I_\tau=\tau_1\alpha_i$.

Remark 3.2. For synthesized system (S_τ), ($T_\tau-E_n$) is non-singular. Simulation results suggest that for proper choices of $\tau_1>1$, $|\tau_2|<1$, Assumption 2.1 holds as well.

From Remarks 3.1 and 3.2 and Lemma 3.1, we obtain

Theorem 3.1. With $\tau_1>1$ and τ_2 sufficiently large, $|\tau_2|<1$, the synthesized LSSM system

$$x(k+1)=T_\tau x(k)+I_\tau, \qquad k=0,1,2,\ldots \tag{S_τ}$$

with constraints

$-1\leq x_i\leq 1$, $i=1,\ldots,n$

has the following properties:

1) Each $\alpha \in B^n \cap L_a$ is an asymptotically stable equilibrium point of (S_τ). In particular, this is true for $\alpha_1,\dots,\alpha_m$.

2) There are no oscillatory solutions.

Remark 3.3. By Theorem 2.4, each $\alpha \in L_a \cap B^n$ is a local minimum of the energy functuon E: $D^n \rightarrow R$, $E(x)=-x^T(T_\tau - E_n)x-2x^T I_\tau$. In fact, it can be proved as in [7] that each $\alpha \in L_a \cap B^n$ is a global minimum of the energy function E.

In the following, we address spurious asymptotically stable equilibrium points of (S_τ).

Lemma 3.2. For a given $\tau_1>1$, if $-1<\tau_2'<\tau_2''<1$ and if $\alpha \in B^n$ is an asymptotically stable equilibrium point of system ($S_{\tau''}$), then α is also an asymptotically stable equilibrium point of system ($S_{\tau'}$), τ' and τ'' are determined by (τ_1,τ_2') and (τ_1,τ_2''), respectively.

We conclude our discussion with the synthesis procedures given below.

Synthesis Procedure 3.1. Suppose we are given m vectors $\alpha_1,\dots,\alpha_m$ in B^n, which are to be stored as asymptotically stable equilibrium points for an n dimensional system (S). We proceed as follows:

1) Compute the $n\times(m-1)$ matrix

$$Y=[\alpha_1-\alpha_m,\dots,\alpha_{m-1}-\alpha_m]. \tag{3.5}$$

2) Perform a singlar value decomposition of Y and obtain the matrices U, V and Σ such that $Y=U\Sigma V^T$, where U and V are unitary matrices and where Σ is a diagonal matrix with the singular values of Y on its diagonal. (This can be accomplished by standard computer routines). Let

$$Y=[y_1,\dots,y_{m-1}], \tag{3.6a}$$

$$U=[u_1,\dots,u_n], \tag{3.6b}$$

and

$$k=\text{dimension of Span}(y_1,\dots,y_{m-1}). \tag{3.6c}$$

From the properties of singular value decomposition we know that k=rank of Σ, $\{u_1,\dots,u_k\}$ is an orthonormal basis of Span$(y_1,\dots,y_{m-1})$ and $\{u_1,\dots,u_n\}$ is an orthonormal basis of R^n.

3) Compute

$T^+=[T_{ij}^+]=\Sigma_{i=1}^k u_i u_i^T$, and

$T^-=[T_{ij}^-]=\Sigma_{i=k+1}^n u_i u_i^T$.

4) choose τ_1, τ_2 and compute

$T_\tau=\tau_1 T^+ - \tau_2 T^-$, and $I_\tau=\tau_1\alpha_m - T_\tau \alpha_m$.

Then all vectors in $L_a \cap B^n$, where L_a=Aspan$(\alpha_1,\dots,\alpha_m)$, including $\alpha_1,\dots,\alpha_m$, will be stored as asymptotically stable equilibrium points in the LSSM system

$$x(k+1)=T_\tau x(k)+I_\tau, \qquad k=0,1,2,\dots \tag{S_τ}$$

with constraints

$-1\le x_i \le 1$, $i=1,\dots,n$.

5) With τ_1 fixed, choose the parameter τ_2 as large as possible, The set of the asymptotically stable equilibrium points contained in B^n is approximately equal to $L_a \cap B^n$.

Remark 3.4. If we wish that in the synthesized system (S_τ) the constant vector $I_\tau=0$, we can modify Synthesis Procedure 4.1 as follows:

a) In step 1), take $Y=[\alpha_1,\dots,\alpha_m]$.

b) In step 4), take $I_\tau=0$.

Then all conclusions will remain unchanged. In particular, each $-\alpha_i$, $i=1,\dots,m$, will also be an asymptotically stable equilibrium point of the synthesized system (S_τ) and the number of elements in $L_a \cap B^n$ will approximately be doubled.

By Theorem 3.1 and Theorem 3.2, Synthesis Procedure 3.1 can be generalized to store data given in $\Delta=\{x\in D^n: x_i=1, -1 \text{ or } 0, 1\leq i\leq n\}$. We have

Synthesis Procedure 3.2. Suppose that we are given m vectors $\alpha_1,\dots,\alpha_m$ in Δ, which are to be stored as asymptotically stable equilibrium points for an n dimensional system (S). We proceeed as follows:

1) For $\alpha, \alpha' \in \{\alpha_1,\dots,\alpha_m\}$, $\alpha' \neq \alpha$, according to Theorem 2.2, the two vectors α and α' can not be stored as asymptotically stable equilibrium points of system (S_τ), if either $\alpha \in \overline{C(\alpha')}$ or $\alpha' \in \overline{C(\alpha)}$, where

$$C(\alpha)=\{x=(x_1,\dots,x_n)^T \in R^n: |x_i|<1 \text{ if } \alpha_i=0, \text{ and } x_i=\alpha_i \text{ if } \alpha_i \neq 0\} \tag{3.7a}$$

$$C(\alpha')=\{x=(x_1,\dots,x_n)^T \in R^n: |x_i|<1 \text{ if } \alpha_i'=0, \text{ and } x_i=\alpha_i' \text{ if } \alpha_i' \neq 0\}. \tag{3.7b}$$

2) Compute the $n\times(m-1)$ matrix

$$Y=[\alpha_1-\alpha_m,\dots,\alpha_{m-1}-\alpha_m].$$

3) Perform a singular value decomposition of Y and obtain the matrices U, V and Σ such that $Y=U\Sigma V^T$, where U and V are unitary matrices and where Σ is a diagonal matrix with the singular values of Y on its diagonal. Let $Y=[y_1,\dots,y_{m-1}]$, $U=[u_1,\dots,u_n]$, and k=dimension of $\mathrm{Span}(y_1,\dots,y_{m-1})$. From the properties of singular value deconmposition, we know that k=rank of Σ, $\{u_1,\dots,u_k\}$ is an orthonormal basis of $\mathrm{Span}(y_1,\dots,y_{m-1})$ and $\{u_1,\dots,u_n\}$ is an orthonormal basis of R^n.

4) Compute

$$T^+=[T_{ij}^+]=\Sigma_{i=1}^{k} u_i u_i^T, \text{ and}$$

$$T^-=[T_{ij}^-]=\Sigma_{i=k+1}^{n} u_i u_i^T.$$

5) Choose τ_1, τ_2 and compute

$$T_\tau=\tau_1 T^+ - \tau_2 T^-, \text{ and } I_\tau=\tau_1\alpha_m - T_\tau \alpha_m.$$

Then all vectors in $L_a\cap\Delta$, where $L_a=\mathrm{Aspan}(\alpha_1,\dots,\alpha_m)$, including $\alpha_1,\dots,\alpha_m$, will be stored as equilibrium points in the LSSM system

$$x(k+1)=T_\tau x(k)+I_\tau, \qquad k=0,1,2,\dots \tag{S_τ}$$

with constraints

$$-1\leq x_i\leq 1, \ i=1,\dots,n.$$

6) For an $\alpha\in\{\alpha_1,\dots,\alpha_m\}$, if $\alpha\in B^n$, α is an asymptotically stable equilibrium point of (S_τ). If $\alpha\notin B^n$, whether α is an asymptotically stable equilibrim point of (S_τ) or not can be determined using Theorem 2.1.

IV. EXAMPLES

In the present section, we demonstrate the applicability of the results of the previous sections.

Example 4.1 In this example, we illustrate use of the Synthesis procedure 3.1. The dimension of the system to be considered is n=10 and m=6 , with

$$\alpha_1=\begin{pmatrix}-1\\1\\-1\\1\\1\\1\\-1\\1\\1\\1\end{pmatrix},\quad \alpha_2=\begin{pmatrix}1\\1\\-1\\-1\\1\\-1\\1\\-1\\1\\1\end{pmatrix},\quad \alpha_3=\begin{pmatrix}-1\\1\\1\\1\\-1\\-1\\1\\-1\\1\\-1\end{pmatrix},\quad \alpha_4=\begin{pmatrix}1\\1\\-1\\1\\-1\\1\\-1\\1\\1\\1\end{pmatrix},\quad \alpha_5=\begin{pmatrix}1\\-1\\-1\\-1\\1\\1\\1\\-1\\-1\\-1\end{pmatrix},\quad \alpha_6=\begin{pmatrix}-1\\-1\\-1\\1\\1\\-1\\1\\1\\-1\\1\end{pmatrix}.$$

It is desired that these vectors be asymptotically stable equilibrium points of LSSM system(S_τ) with τ_1=1.5 and τ_2=0.5. Our simulation results verified that $\alpha_1,...,\alpha_6$ have been stored as asymptotically stable equilibrium points for the synthesized LSSM system. To save space, the resulting matrix T and vector I are not given here.

System (S_τ) has additional asymptotically stable equilibrium points in D^n-$\{\alpha_1,...,\alpha_6\}$ given by

$$\alpha'_1=\begin{pmatrix}-1\\1\\0\\0\\1\\1\\0\\-1\\1\\-1\end{pmatrix},\quad \alpha'_2=\begin{pmatrix}1\\1\\0\\0\\-1\\1\\0\\-1\\1\\-1\end{pmatrix},\quad \alpha'_3=\begin{pmatrix}-1\\1\\0\\0\\1\\-1\\1\\-1\\1\\0\end{pmatrix},\quad \alpha'_4=\begin{pmatrix}1\\1\\0\\0\\-1\\-1\\1\\-1\\1\\0\end{pmatrix},\quad \alpha'_5=\begin{pmatrix}-1\\1\\0.87\\1\\-1\\1\\-0.30\\-0.70\\1\\-1\end{pmatrix},\quad \alpha'_6=\begin{pmatrix}-1\\1\\-0.87\\1\\1\\-1\\0.30\\0.70\\1\\1\end{pmatrix},$$

$$\alpha'_7=\begin{pmatrix}1\\1\\-0.87\\1\\-1\\-1\\0.30\\0.70\\1\\1\end{pmatrix},\quad \alpha'_8=\begin{pmatrix}-1\\-1\\0.62\\1\\-1\\-1\\1\\-0.21\\-1\\-1\end{pmatrix},\quad \alpha'_9=\begin{pmatrix}1\\-1\\-1\\-0.49\\1\\-1\\1\\0.06\\-1\\1\end{pmatrix},\quad \alpha'_{10}=\begin{pmatrix}-1\\-1\\-1\\1\\1\\1\\0\\1\\-1\\0\end{pmatrix},\quad \alpha'_{11}=\begin{pmatrix}1\\-1\\-1\\1\\-1\\1\\0\\1\\-1\\0\end{pmatrix},\quad \alpha'_{12}=\begin{pmatrix}1\\1\\-1\\-1\\1\\1\\0\\-1\\1\\0\end{pmatrix}.$$

Also by Theorem 2.1, we determine that S_τ has 158 unstable equilibrium points.

Example 4.2 In order to ascertain how typical the results of Example 4.1 are, we evaluate the number of spurious states found by varying the parameters τ_1 and τ_2 in Example 4.1.

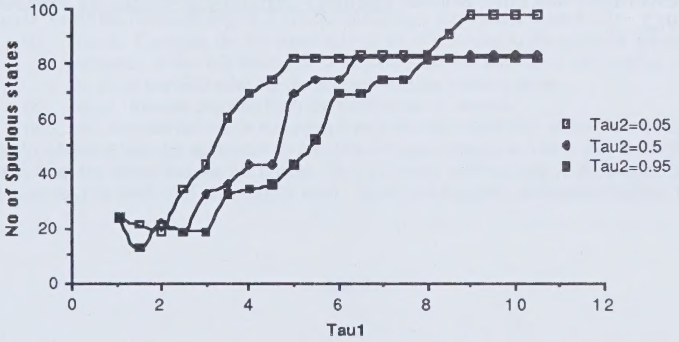


Figure 4.1

Remark 4.1 By the results shown in the Figure 4.1, we see that

1) Using the present method, the $\alpha_1,\ldots,\alpha_6$ are always stored as stable output vectors. Also as τ_2 increases($-1<\tau_2<1$), the number of spurious stable output vectors decreases.

2) In view of these simulation results, the present synthesis method appears to offer improvements over some of the existing methods(e.g. Hopfield's and Personnaz's methods) in the following ways :

i) The speed of convergence in digital simulations is highly improved.

ii) The present model guarantees to store all desired patterns as asymptotically stable equilibria.

iii) Given the dimension of the system n, the network has the ability to store more patterns compared to other design methods..

iv) We can estimate the domain of attraction of the asymptotically stable equilibria by determining the unstable equilibrium points.

REFERENCES

[1] J. J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties like Those of Two-state Neurons", Proc. Nat'l. Acad. Sci., U.S.A., Vol. 81, May 1984, pp. 3088-3092.

[2] J. J. Hopfield and D.W. Tank, "Neural Computation of Decisions in Optimization Problems", Biological Cybernectics, Vol 52, 1985, pp. 141-152.

[3] D. W. Tank and J.J. Hopfield, " Simple 'Neural' Optimization Networks: an A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", IEEE Trans. On Circuits and Systems, Vol. CAS-33, No.5, May 1986, pp. 533-541.

[4] L. Personnaz, I. Guyon and G. Dreyfus, "Information Storage and Retrieval in Spin-glass like Neural Networks", J. Physique Letters, Vol. 46, April 1985, pp. L359-L365.

[5] L. Personnaz, I. Guyon and G. Dreyfus, "Collective Computational Properties of Neural Networks: New Learning Mechanisms", Physical Review A, The American Physical Society, Vol. 34, No. 5, Nov. 1986, pp. 4217-4228.

[6] J. Li, A. N. Michel and W. Porod , "Qualitative Analysis and Synthesis of a Class of Neural Networks", IEEE Trans. on Circuites and Systems, Vol. 35, No. 8, Aug. 1988, pp. 976-986.

[7] J. Li, A. N. Michel and W. Porod, "Qualitative Analysis and Systhesis of a Class of Neural Netwoks: Variable Structure Systems with Infinite Gains", IEEE Trans. on Circuits and Systems, Vol. 36, May 1989, pp. 713-731.

[8] J. Li, A. N. Michel and W. Porod, "Analysis and Synthesis of a Class of Neural Networks : Linear Systems Operating in Saturated Modes", Proceedings of the 1989 IEEE International Symposium on Circuts and Systems, Portland, Oregon, May 1989, pp. 830-833.

[9] A. N. Michel, J. A. Farrel and H. F. Sun, "Analysis and Synthesis Techniques for Discrete Time Neural Network Models", Proceedings of the 1989 IEEE Conference on Decision and Control, Tampa, Florida, Dec. 1989 (to appear).

[10] A.N. Michel and C.J. Herget, Mathematical Foundations in Engineering and Science, Prentice-Hall, Englewood Cliffs, N.J., 1981.

[11] R.K. Miller and A.N. Michel, Ordinary Differencial Equations, Academic Press, New York, 1982.

ARTIFICIAL NEURAL NETWORKS IN THE MATCH PHASE OF RULE-BASED EXPERT SYSTEMS

MICHAEL A. SARTORI, KEVIN M. PASSINO, and PANOS J. ANTSAKLIS
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

In rule-based expert systems, it is often the case that the left-hand-sides of the rules must be repeatedly compared to the contents of some “working memory”. Normally, the intent is to determine which rules are relevant to the current situation (i.e., to find the “conflict set”). The traditional approach to solve such a “match phase problem” for production systems is to use the Rete Match Algorithm. Here, an artificial neural network is used to solve the match phase problem for rule-based expert systems. A syntax for premise formulas (i.e., the left-hand-sides of the rules) is defined, and working memory is specified. From this, it is shown how to form an artificial neural network (here, the multi-layer perceptron) that finds all of the rules which can be executed for the current situation in working memory. The conventional Rete Match Algorithm approach is compared to the novel approach presented here.

1 INTRODUCTION

In rule-based expert systems, the problem of finding which rules are executable from a given set of rules at different instances in time is often encountered. At each time instant, the left-hand-side of every rule of a given set of rules is compared to the dynamically changing "working memory" of the system. The rules whose left-hand-sides are satisfied by the current working memory form the "conflict set" at that particular time. The problem of determining the conflict set from the contents of working memory at a particular time is referred to here as the *match phase problem.* Typically, in such rule based systems, an "inference engine" or some other suitable algorithm finds the rules which are executable (in the "match phase"), chooses one (in the "select phase"), and executes it (in the "act phase"). This paper focuses on solving the match phase problem via an artificial neural network. The problem of choosing which rule to execute (e.g., via an inference engine using certainty factors or fuzzy logic) and the problem of deciding the manner in which the chosen rule is to be executed are not addressed here. A brief overview of where the match phase problem is found in rule-based expert systems is now given and is followed by a brief description of the Rete Match Algorithm, the conventional solution to the match phase problem.

A rule-based expert system uses *rules*, sometimes referred to as *productions* or *production rules*, to represent knowledge and uses an *inference engine* to perform the actions of the expert system. In general, a rule is of the form:

IF (antecedent) THEN (consequence) (1)

Customarily, the antecedent is referred to as the left-hand-side, and the consequence is referred to as the right-hand-side. The *working memory* (*data memory*) is a dynamically changing memory which contains the data that is compared to the left-hand-side of the rules. The individual elements of the working memory are referred to as the *working memory elements.* The inference engine performs the comparison of the working memory elements to the left-hand-sides of the rules, chooses which rules are executable for the given state of the expert system, chooses one of the executable rules, and executes it. Often the inference engine is viewed as having a three phase cycle [1,2]:

(i) *Match.* Compare the left-hand-side of all of the rules to the working memory elements. If the left-hand-side is satisfied, include the rule in the *conflict set*, the set of satisfied rules for the present working memory state.
(ii) *Select.* Choose one rule from the conflict set to execute.
(iii) *Act.* Execute the rule in accordance with the right-hand-side of the chosen rule.

The results obtained here are applicable to rule-based expert systems in which the match phase can be separated from the select and the act phases. It is assumed, without loss of generality, that forward chaining instead of backward chaining is used. Rule-based expert systems developed with OPS5, EMYCIN, ROSIE, and KEE as cited in [1] and Level 5 as described in [3] may benefit from an artificial neural network implementation of the match phase as described in this paper.

The conventional solution to the match phase problem (particularly for rule-based expert systems) utilizes the Rete Match Algorithm [4]. If the expert system's rules are explicitly of the form depicted in equation (1) and if the inference engine explicitly follows the three phase cycle, the rules are referred to as "productions" and the inference engine is referred to as the "production interpreter". Of the three phases, the match phase traditionally consumes the most time of the production interpreter. Using conventional approaches, a production interpreter can spend more than 90% of its time in the match phase of the cycle [4]. The Rete Match Algorithm, introduced in [4,5] and implemented in the OPS5 expert system building tool in [1], avoids the brute force approach by manipulating the rules and the working memory elements to form a software tree structure to increase the speed of the interpreter. The time needed to perform the match phase using the Rete Match Algorithm can be quantified. For a production system, if W is the number of working memory elements in the working memory and A is the number of atomic propositions per rule, the effect of the working memory size on the time for one firing using the Rete Match Algorithm is $O(1)$ for the best case and $O(W^{2A-1})$ for the worst case [5] (where the “O” notation denotes “on the order of” and is the standard one defined in [6]). If R is the number of rules in the production system, the effect of the number of rules on the time for one firing using the Rete Match Algorithm is $O(\log_2 R)$ for the best case and $O(R)$ for the worst case [5]. So, the time for one firing of the match phase using the Rete Match Algorithm is dependent on both the number of rules and the number of working memory elements. In the following, it will be shown that when the proposed artificial neural network solution to the match phase problem is used, the processing time is independent of both the number of rules and the number of working memory elements.

Since the Rete Match Algorithm's introduction, other Rete based algorithms which attempt to increase the speed of the interpreter have been introduced [7-14]. To further reduce the amount of time consumed by the interpreter in the match phase, special hardwares have been developed using parallelism and multiprocessor architectures [10,15-23]. The Rete Match Algorithm has also been implemented on a multiprocessor machine, and an increase in speed was reported for specific rule-based systems [24]. Almost all of these attempts are based, at least in part, on the Rete Match Algorithm, which is assumed to yield the most efficient match phase for a production interpreter. These architectures, as reported in the literature, strive to decrease the time required in the match phase by attempting to match as many rules as possible in parallel and by attempting to fire as many rules as possible in parallel. Using the YES/OPS production system language and advances in the Rete Match Algorithm, a drop in CPU time was reported in [7]. Using partitioning of the productions, a reduction in production cycles was reported in [11,12,14]. The results from the literature of using special hardware are summarized in Table I. The current hardware implementations cited in Table I use either many processors or many processing elements.

Table I. Summarized results of special hardware implementations.

Architecture	WME Changes/Sec	Production Cycles/Sec	Hardware Needed	Reference
CMU-PMS	9400	-	32 Ps	[10]
DADO	215	85	1023 PEs	[21,10]
MAPPS	10,000	-	128 PEs	[18]
NON-VON	2000	903	16032 PEs	[18]
Olfazer's	4500	-	512 Ps	[10]
PESA-1	25,000	8000	32 PEs	[19]

WME = working memory element
- = not available
Ps = processors
PEs = processing elements

The method proposed here for the match phase is quite different from the Rete Match Algorithm and the above mentioned modifications; the approach described in this paper simultaneously matches all of the rules to all of the working memory elements in parallel via an artificial neural network and, as a result, produces the conflict set. Artificial neural networks are chosen here because of their ability to perform fast computations in a massively parallel fashion. The approach to solving the match phase problem with an artificial neural network begins by first defining "premise formulas" which represent the left-hand-sides of the rules. The match phase problem is defined in Section 2 as the determination of the truth of the premise formulas for the current working memory. The artificial neural network that is used here to implement the match phase is the multi-layer perceptron. In Section 3, a procedure for designing a multi-layer perceptron for a single premise formula is described and extended to one for designing a multi-layer perceptron for a set of given premise formulas. The input to the multi-layer perceptron is the working memory, and the output is the conflict set. The specially designed multi-layer perceptron simultaneously finds all of the rules which are executable at any particular time by matching all of the rules in parallel to the current working memory. The use of an artificial neural network as the match phase in the three phase cycle of a rule-based expert system is further explained in Section 3. Finally in Section 4, concluding remarks are made including a citing of some of the potential limitations in using the artificial neural network solution for the match phase. Note that this work is an extension of the research reported in [25] via the results in [26,27]. The artificial neural network used in the previous work was the model termed the "ProNet", which was obtained by modifying the Hamming net of [28]. The multi-layer perceptron was used rather than the ProNet because the multi-layer perceptron is better able to address a greater diversity of types of rules as well as rules which contain real numbers.

2 THE MATCH PHASE PROBLEM

In this section, the premise formulas for the left-hand-sides of the expert system's rules are defined, and the match phase phase problem is defined in terms of the premise formulas and the working memory. The premise formulas correspond to the left-hand-sides of the expert system's rules. Rule-based expert systems with left-hand-sides which can be described using the premise formulas described below can have the match phase of their inference engine performed by an artificial neural network as discused in the following sections.

The syntax defining the premise formulas is now described. Let $|\mathbf{S}|$ denote the number of elements in the set $\mathbf{S}$. Let $\mathbf{A} = \{a_1, a_2, ..., a_n\}$ be a nonempty finite set of *atomic propositions* a_i, where $|\mathbf{A}| = n$. Such propositions will represent facts that are stored in working memory. Let $\mathbf{X} \subset \mathbb{R}^p$, where $\mathbb{R}$ denotes the set of real numbers. Each $\mathbf{x} \in \mathbf{X}$ represents numeric information in working memory as discussed below. The *premise formulas* are composed of the following symbols:

(a) Atomic propositions $a \in \mathbf{A}$,

(b) Real numbers x_i for $1 \leq i \leq p$, where $\mathbf{x} \in \mathbf{X}$ and $\mathbf{x} = [x_1, x_2, ..., x_p]^t$ (and "t" denotes transpose),

(c) The Boolean connectives: $\neg$ (negation), $\wedge$ (disjunction), and $\vee$ (conjunction),

(d) The set $\mathbb{R}$ of real numbers,

(e) The predicate symbols: > (greater than), < (less than), ≥ (greater than or equal to), and ≤ (less than or equal to), and

(f) The parentheses: (and).

The rules for forming premise formulas are as follows:

(1) A single atomic proposition $a \in \mathbf{A}$ is a premise formula.

(2) If σ is a premise formula, then $\neg\sigma$ is a premise formula.

(3) If σ and ψ are premise formulas, then so are $(\sigma \wedge \psi)$ and $(\sigma \vee \psi)$.

(4) If $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} = [x_1, x_2, ..., x_p]^t$, and $r \in \mathbb{R}$, then $(x_i > r)$, $(x_i < r)$, $(x_i \geq r)$, and $(x_i \leq r)$ for any i such that $1 \leq i \leq p$ are premise formulas.

(5) Nothing is a premise formula unless it can be obtained by finitely many applications of (1) - (4) above.

If some left-hand-side of a rule has either the Boolean connective $\Rightarrow$ (implication) or $\Leftrightarrow$ (equivalence), then the following substitutions can be made. If σ and ψ are premise formulas, then $(\sigma \Rightarrow \psi)$ can be replaced by the equivalent premise formula $(\neg\sigma \vee \psi)$, and $(\sigma \Leftrightarrow \psi)$ by the equivalent premise formula $((\sigma \wedge \psi) \vee (\neg\sigma \wedge \neg\psi))$. If some left-hand-side of a rule has either the predicate symbol = (equal) or ≠ (not equal), then the following substitutions can be made. If $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} = [x_1, x_2, ..., x_p]^t$, and $r \in \mathbb{R}$, then $(x_i = r)$ can be replaced by the equivalent premise formula $((x_i \geq r) \wedge (x_i \leq r))$, and $(x_i \neq r)$ by the equivalent premise formula $((x_i > r) \vee (x_i < r))$ for $1 \leq i \leq p$.

Assume that the working memory for the rule-based system is composed of atomic propositions $a \in \mathbf{A}$ and the current working memory information $\mathbf{x} \in \mathbf{X}$. The output of the working memory is specified next. Define the function

$$V{:}A \rightarrow \{0,1\} \tag{2}$$

where $V(a) = 1$ indicates "a is true" and $V(a) = 0$ indicates that "a is false". Let the n-vector $\mathbf{v_k} = [V(a_1), V(a_2), ..., V(a_n)]^t$ with components $V(a_i)$ representing the truth values of all of the atomic propositions at step k. Let $\mathbf{x_k} \in \mathbf{X}$ denote the value of $\mathbf{x}$ at step k. At step k, the output of the working memory is defined to be the (n+p)-vector $\mathbf{u_k} = [\mathbf{v_k}^t, \mathbf{x_k}^t]^t$. Let $\mathbf{U}$ denote the set of all possible working memory outputs.

Let $\Phi = \{\phi_1, \phi_2, .., \phi_m\}$ denote a finite set of premise formulas. Let $\mathbf{Y} \subset \{0,1\}^m$. Number the rules in the rule base from 1 to m. For the i^{th} rule in the rule base form the premise formula ϕ_i and associate ϕ_i with a component y_i of $\mathbf{y_k} = [y_1, y_2, ..., y_m]^t$ for $1 \le i \le m$. If $y_i = 1$ for $1 \le i \le m$, then the premise formula ϕ_i is true, and the i^{th} rule is executable and included in the conflict set. If $y_i = 0$, then the corresponding rule is not included in the conflict set. Hence, at step k, $\mathbf{y_k} \in \mathbf{Y}$ represents the conflict set.

The match phase problem can be solved by implementing the function

$$P{:}U \times \Phi \rightarrow Y \tag{3}$$

with a multi-layer perceptron. The utilization of the multi-layer perceptron to perform the match phase in a rule-based expert system is illustrated in Figure 1. The input to the multi-layer perceptron, which describes the working memory, is the vector $\mathbf{u_k}$. The input vector is compared to the left-hand-sides of the entire set of rules described by the set Φ which is stored in the multi-layer perceptron as its weights, biases, and interconnections. The output of the multi-layer perceptron is the vector $\mathbf{y_k}$, which denotes the conflict set. Thus, the multi-layer perceptron specifies the conflict set $\mathbf{y_k} \in \mathbf{Y}$ at each time instant k for all possible inputs $\mathbf{u_k} \in \mathbf{U}$ and premise formulas $\phi \in \Phi$, and hence implements the function P.

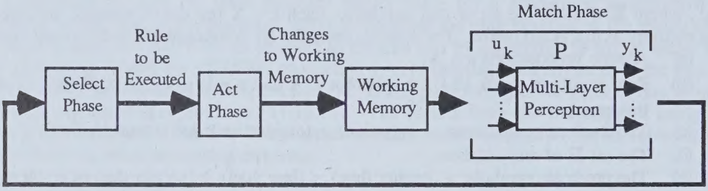


Figure 1 The multi-layer perceptron implementation of the match phase in a rule-based expert system.

Following Figure 1, from the output vector $\mathbf{y_k}$, the conflict set is placed in a form which is usable by the select phase. The select phase can be implemented in numerous ways; for an explanation of some of the possibilities for accomplishing this, see [1,2,29]. Once the rule is chosen by the select phase, it is executed via the act phase. The execution of a rule causes changes to the working memory. The vector $\mathbf{v_{k+1}}$ is produced by altering the vector $\mathbf{v_k}$ such that some of the elements $V(a_i)$ are changed form one to zero (indicating that a_i becomes false) and others from zero to one (indicating that a_i becomes true). The vector $\mathbf{x_k}$ can also be changed according to the fired rule to produce the vector $\mathbf{x_{k+1}}$. So, the working memory is updated, and the input vector $\mathbf{u_k}$ is changed to $\mathbf{u_{k+1}}$. The new input vector $\mathbf{u_{k+1}}$ is used as an input to the multi-layer perceptron which gives the output $\mathbf{y_{k+1}}$, and the process is repeated. Next, it is shown how to construct a multi-layer perceptron which implements the function P and hence solves the match phase problem.

3 AN ARTIFICIAL NEURAL NETWORK SOLUTION TO THE MATCH PHASE PROBLEM

Artificial neural networks are used to perform computations in a massively parallel fashion. They are processing models which derive their structure and functionality as an interconnected network of neurons from models of biological neurons. Each neuron of the artificial neural network has many inputs and one output. The output of each neuron is generally considered to be the weighted sum of its inputs passed through a nonlinear function. The output is then used as inputs to other neurons. Artificial neural networks are characterized by the type of neurons used, the way in which the weights are selected, and the types of interconnections that are allowed between neurons. Here, a specific artificial neural network called the *multi-layer perceptron* is used as the model for solving the match phase problem. The multi-layer perceptron is a feed forward model which is restricted here so that it is partially connected and also so that it does not self-adjust its weights or learn.

Instead of using the multi-layer perceptron, other neural network models may be used for the match phase problem, but these do not appear to be as favorable. For example, the Hopfield network may be considered [30], but because it uses feedback to determine its output, it requires time to converge to a minimum; the multi-layer perceptron always determines its outputs immediately since it is a feed forward network with theoretically no significant delays. Also, if the weights for the Hopfield network are not chosen properly, spurious states may result. Artificial neural networks which use unsupervised training, such as the Kohonen network [31], may not be as useful as the multi-layer perceptron since the input and output relationships for the artificial neural network are known *a priori*, and thus the clustering methodology of the unsupervised training is not needed. Artificial neural networks which use supervised training, such as the Back Propagation Training Algorithm applied to the multi-layer perceptron [32], also may not be as useful since the weights for the network can be selected as explained in Section 3.1., and so the training of the weights of the multi-layer perceptron is not required. In addition, problems which occur in learning algorithms which adjust the weights do not occur for the weight selection procedure proposed here. For example in the Back-Propagation Training Algorithm, the convergence of the weights does not always occur [30]. For these reasons, the multi-layer perceptron was judged as the most suitable artificial neural network model for use in the match phase. This brief discussion about artificial neural networks and the various neural network models is not meant to be encompassing; the interested reader may examine [30] for further information on a number of different artificial neural networks used for other applications.

3.1 The Multi-Layer Perceptron

The multi-layer perceptron is a feed forward artificial neural network considered here to contain at least one *hidden layer* between the *input* and the *output layers*. A multi-layer perceptron with one hidden layer and with two nodes in each layer is illustrated in Figure 2. The output of one layer is cascaded to the input of the next one. In general, the input layer feeds the first hidden layer, and the last hidden layer feeds the output layer. The input of the multi-layer perceptron is applied to the input layer. The input to the multi-layer perceptron considered here is the vector **u** with components u_i, which contains (n+p) continuous real valued elements. The vector **z'** with components z'_i, which contains q binary elements, is the output of the input layer and the input to the hidden layer. The vector **z''**, which contains r binary elements, is the output of the hidden layer and the input to the output layer. The vector **y** with components y_i, which contains m binary elements, is the output of the multi-layer perceptron.

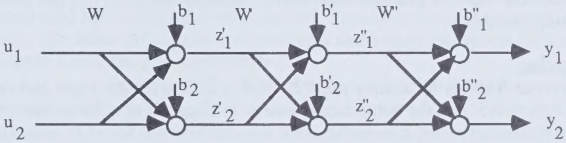
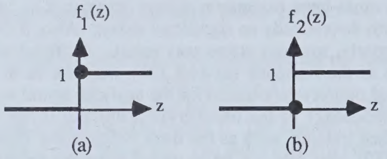


Figure 2 The multi-layer perceptron for two nodes at each layer.

In Figure 2, the *nodes* are denoted with circles and the *biases* with arrows that point downward. The biases on the input layer are denoted by b_i, on the hidden layer by b'_i and on the output layer by b''_i. The *weights* are denoted by all of the other arrows (which are labelled with the weights) that are between **u** and the input nodal layer, between **z'** and the hidden nodal layer, and between **z''** and the output layer. Note that unlike the traditional three layer perceptron used with the Back Propagation Algorithm, the input layer can assume non-unity valued weights. The element w_{ij} of the $(n+p) \times q$ matrix W denotes the weight on the arc from u_i to the node with z'_j as its output. The $q \times r$ matrix W' denotes the weights on the arcs from each z'_i to the node with z''_j as its output. The $r \times m$ matrix W'' denotes the weights on the arcs from each z''_i to the node with y_j as its output. The weights on the arcs connecting the output nodal layer to the outputs y_i are unity. For convenience, if the weight of any arc is zero the arc will be omitted from the graphical representation of the perceptron, and if the weight of any arc is unity, the arc will be represented with no weight denoted.

Each node produces at its output a summation of its weighted inputs and its bias which is passed through a threshold nonlinearity. The result is a binary output for each layer. Two typical threshold nonlinearities are illustrated in Figure 3. Note that in this paper, nodes which use the threshold of Figure 3(a) are unshaded and those which use the threshold of Figure 3(b) are shaded.



(a) (b)

Figure 3 Threshold nonlinearities for the multi-layer perceptron.

The input to the jth threshold nonlinearity of the input layer, denoted by $\tilde{z}_j(k)$, is the weighted sum of the inputs added to the bias

$$\tilde{z}_j(k) = \left(\sum_{i=1}^{M} w_{ij} u_i(k) \right) + b_j. \tag{4}$$

If f_{tj} denotes a threshold nonlinearity of type t (where here $t = 1$ or 2 following Figure 3) for the jth node, then the output of the input nodal layer is given by

$$z'_j(k) = f_{tj}(\tilde{z}_j(k)). \tag{5}$$

The outputs of the hidden layers and the output layer are given by similar relations. With these equations the input-output relationship for the multi-layer perceptron is specified. Using this description of the multi-layer perceptron, the technique to determine the weights, biases, and number of nodes for the multi-layer perceptron used in solving the match phase problem is given next.

3.2 Construction of the Multi-Layer Perceptron

The construction of a multi-layer perceptron for one premise formula is examined first. Given a single premise formula $\phi \in \Phi$, an artificial neural network is constructed which indicates whether or not ϕ is true for any situation in working memory $\mathbf{u}_k \in \mathbf{U}$. The construction of the multi-layer perceptron has two steps: forming the nodes and connecting the nodes. The nodes can be one of three types, each of which corresponds to a different premise formula type (i.e., (2), (3), and (4) in the rules for forming premise formulas).

Negation of Formulas:

For a premise formula formed by rule (2), form a node with one input and one output. The weight is -1, the bias is 0.5, and the nonlinearity used is in Figure 3(a). The network for the premise formula

$$\phi_a = \neg\sigma \tag{6}$$

is shown in Figure 4(a), where σ is a premise formula.

Disjunction and Conjunction of Formulas:

For a premise formula defined by rule (3), let σ_i be a premise formula for i = 1, ..., j, and consider premise formulas either of the type $(\sigma_1 \wedge \sigma_2 \wedge \ldots \wedge \sigma_j)$ or $(\sigma_1 \vee \sigma_2 \vee \ldots \vee \sigma_j)$. Form a node with j inputs and 1 output. The weights are unity and the nonlinearity used is in Figure 3(a). If the Boolean connector is $\wedge$, then the bias of the node is $-(j - 0.5)$ where j is the number of premise formulas in the conjunction. If the Boolean connector is $\vee$, then the bias of the node is -0.5 no matter how many formulas are in the disjunction. The network formed for the premise formula

$$\phi_b = ((\sigma_1 \wedge \sigma_2) \vee \sigma_3) \tag{7}$$

is shown in Figure 4(b), where σ_1, σ_2, and σ_3 are premise formulas.

Relational Formulas:

For a premise formula formed by rule (4) of the syntax (e.g., $(x_i > r)$, $(x_i < r)$, $(x_i \geq r)$, and $(x_i \geq r)$ for $1 \leq i \leq p$, where $\mathbf{x} = [x_1, x_2, \ldots, x_p]^t$ and $r \in \mathbb{R}$) form a node with one input and one output. Connect the node's input to the appropriate element of the vector **x** in the input vector **u** of the multi-layer perceptron. If the predicate symbol is > or ≥, then the weight is 1 and the bias is -r. If the predicate symbol is < or ≤, then the weight is -1 and the bias is r. If the predicate symbol is ≤ or ≥, use the nonlinearity of Figure 3(a). If the predicate symbol is < or >, use the nonlinearity of Figure 3(b). The network for the premise formula

$$\phi_c = ((x_1 \geq 2) \vee (x_2 < -4)) \tag{8}$$

is shown in Figure 4(c), where $\mathbf{x} \in \mathbf{X}$, $\mathbf{x} = [x_1, x_2]^t$.

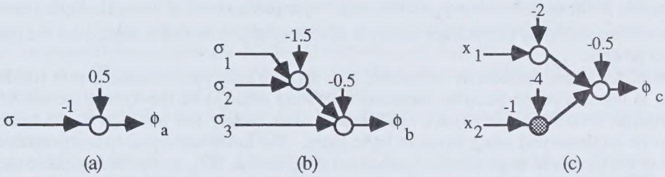


Figure 4 The multi-layer perceptron for premise formulas (a) $\phi_a = \neg\sigma$, (b) $\phi_b = ((\sigma_1 \wedge \sigma_2) \vee \sigma_3)$, and (c) $\phi_c = ((x_1 \geq 2) \vee (x_2 < -4))$.

Once the nodes of the multi-layer perceptron are formed, the connections between them are specified by an inductive process. The output of the node describing the outermost parentheses of the premise formula is the output of the multi-layer perceptron. Let this node be the first layer (i.e., the output layer). The connections between the first layer and the second layer (i.e., the last hidden layer) are made between the inputs of the first layer's node and the outputs of the nodes formed for the premise formulas of the second layer. If the first layer's node requires an element of the input vector as an input, the appropriate connection between the input vector and the node's input is made. Next, the connections between the first and second layer and the third layer (i.e., the second to the last hidden layer) are made. If the second layer's node requires an element of the input vector as an input, the appropriate connection between the input vector and the node's input is made. This process continues for each successive layer until all inputs for all nodes are connected. Notice that given any premise formula, the method used here to form the nodes and the interconnections of the multi-layer perceptron can be mechanized. The entire process for constructing a multi-layer perceptron for a premise formula is illustrated in Figure 5 with the premise formula

$$\phi_9 = ((a_1 \wedge (x_1 < r_1)) \vee \neg(a_1 \wedge a_2) \vee \neg a_1 \vee a_3) \tag{9}$$

where $\mathbf{A} = \{a_1, a_2, a_3\}$ and $\mathbf{x} = [x_1]$.

Using the steps detailed above to produce a neural network for an individual premise formula, the multi-layer perceptron for every premise formula $\phi \in \Phi$ is formed by repeating the above procedure for every premise formula. If two or more premise formulas share a similar premise formula, the node for that premise formula can be shared by the other premise formulas. Thus, one combined multi-layer perceptron implements all of the premise formulas in the set Φ. Notice that the possibility exists for automating the development of the multi-layer perceptron once the set Φ of premise formulas is given.

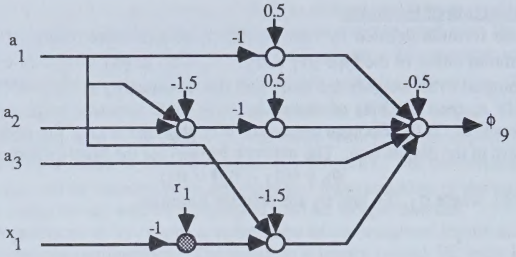


Figure 5 The multi-layer perceptron for the premise formula of equation (9).

As stated in Section 1, the time for the multi-layer perceptron to process the match phase is independent of both the number of working memory elements and the number of rules. Using the artificial neural network solution to the match phase problem, the working memory size does not affect the time required to perform the match phase. This is the case because the working memory is processed in parallel as the input vector $\mathbf{u_k}$ of to the multi-layer perceptron, and thus the working memory size is not detrimental to the time required to process the match phase. In addition, the number of rules (premise formulas) also does not affect the time required to perform the match phase since the truth value of the rules with regard to the current working memory are represented as the parallel elements in the output vector $\mathbf{y_k}$ of the multi-layer perceptron. If the multi-layer perceptron is physically implemented, only the number of layers adversely affects the time needed for the multi-layer perceptron to process.

The match phase problem as formulated here appears to be implementable with standard logic gates. This is not true since premise formulas following rule (4) of the syntax are allowable. If premise formulas following rule (4) are not allowed, then clearly this solution to the match phase problem can be implemented using Boolean logic gates. The multi-layer perceptron proposed here is also similar to the threshold logic circuits (networks) described in [33], except that in these circuits the theory only allows a finite number of discrete inputs. Such threshold logic networks could however be used to implement the match phase.

4 CONCLUDING REMARKS

In this paper, it was proposed to use an artificial neural network to perform the match phase of rule-based expert systems. Using the proposed approach, the match phase is performed by matching in parallel all of the rules to all of the working memory elements. The result of doing this is the formation of the conflict set for the current situation of working memory. Given a set of rules with left-hand-sides following the prescribed syntax, the construction of a multi-layer perceptron which implements the match phase was described. The multi-layer perceptron match phase approach was compared to a conventional match phase interpreter for production systems, the Rete Match Algorithm. Note that an extended version of the work in this paper appears in [34]. These extensions include a formula for the maximum number of nodes needed to implement any premise formula with a multi-layer perceptron, a procedure for reducing the number of layers in a multi-layer perceptron implementing any premise formula to at most three, and as an example of the proposed match phase approach, the construction of a multi-layer perceptron for the rules of a rule-based expert system in [1]. Next, some potential limitations to the artificial neural network approach are discussed.

Comparing the types of rules used by the artificial neural network match phase to the types of rules used by the Rete Match Algorithm match phase, the artificial neural network approach is able to represent many rules which can be described in OPS5, and thus can be used as the match phase for many production systems which have been described using OPS5. For a description of OPS5 rules, see [1]. The transformation (which is often immediate in most cases) of an OPS5 rule's left-hand-side into the appropriate premise formula is the step required in order to use the artificial neural network approach. One limitation of this approach is that the left-hand-side of an OPS5 rule can not contain certain variables (for example, a symbolic variable which takes on an infinite number of values). This limitation can be overcome, however, if these variables can be redefined so that they take on only a finite number of values.

Another potential limitation of the multi-layer perceptron implementation of the match phase as well as another difference between the two match phase implementations (the multi-layer perceptron and the Rete Match Algorithm) is the procedure for the forming of new rules while the inference process is executing (sometimes referred to as "learning rules" or "automatic knowledge acquisition"). The Rete Match Algorithm uses changes in its software tree structure to represent added rules (and deleted rules). When a new rule is added using the artificial neural networks approach, new arcs, nodes, weights, and biases are augmented to the already existing multi-layer perceptron as cited by the new rule. (In a similar manner, old rules can be deleted by removing the appropriate parts of the multi-layer perceptron.) When a rule is added, the time for the multi-layer perceptron to process may not be affected because the depth of the network may not increase, but the size of the neural network will undoubtedly increase. If a rule needs to be changed, arcs, nodes, weights, and biases associated with the changed rule are affected, and the time and space required by the multi-layer perceptron may increase. If the multi-layer perceptron is implemented with hardware, the adding, changing, or even deleting of a rule could be costly. For this reason, a rule-based expert system which requires a fast match phase should first be developed using software. Then once it is in its final form, the rule-based expert system can be implemented with a multi-layer perceptron to more efficiently utilize the proposed artificial neural network approach.

5 REFERENCES

[1] Brownston L., Farrell R., E. Kant, N. Martin, *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*, Addison-Wesley, Reading, MA, 1985.

[2] Buchanan B.G., Shortliffe E.H., *Rule-Based Expert Systems*, Addison-Wesley, Reading, MA, 1984.

[3] *Level 5 Expert System Software*, Information Builders Incorporated, New York, NY, 1989.

[4] Forgy C.L., *On the Efficient Implementation of Production Systems*, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, February 1979.

[5] Forgy C.L., "Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Problem", *Artificial Intelligence*, Vol. 19, pp. 17-37, 1982.

[6] Aho A.V., Hopcroft J.E., Ullman J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[7] Schor M.I., Daly T.P., Ho S.L., Tibbitts B.R., "Advances in Rete Pattern Matching", *Proceedings of the 1986 National Conference on Artificial Intelligence*, pp. 226-232, 1986.

[8] Stolfo S.J., "Five Parallel Algorithms for Production System Execution on the DADO Machine", *Proceedings of the National Conference on Artificial Intelligence*, pp. 300-307, 1984.

[9] Miranker D.P., "Performance Estimates for the DADO Machine: A Comparison of TREAT and RETE", *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp. 449-457, 1984.

[10] Gupta A., Forgy C.L., Newell A., Wedig R., "Parallel Algorithms and Architectures for Rule-Based Systems", *Proceedings of the 13th International Symposium on Computer Architecture*, pp. 28-37, 1986.

[11] Ishida T., Stolfo S., "Towards the Parallel Execution of Rules in Production System Programs", *Proceedings of the 1985 International Conference on Parallel Processing*, pp. 568-575, 1985.

[12] Oflazer K., "Partitioning in Parallel Processing of Production Systems", *Proceedings of the 1984 International Conference on Parallel Processing*, pp. 92-100, 1984.

[13] Moldovan D.I., "A Model for Parallel Processing of Production Systems", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 568-573, 1986.

[14] Tenorio M.F.M., Moldovan D.I., "Mapping Production Systems into Multiprocessors", *Proceedings of the 1985 International Conference on Parallel Processing*, pp. 56-62, 1985.

[15] Won H., "On Parallel Processing of Universal Rule Based Expert System", *Proceedings of the 1987 Western Conference on Expert Systems*, pp. 136-143, 1987.

[16] Forgy C., Gupta A., Newell A., Wedig R., "Initial Assessment of Architectures for Production Systems", *Proceedings of the 1984 National Conference on Artificial Intelligence*, pp. 116-120, 1984.

[17] Forgy C., Goopta A., "Preliminary Architecture of the CMU Production System Machine", *Proceedings of the Nineteenth Annual Hawaii International Conference on System Sciences*, 1986, pp. 194-200, 1986.

[18] Oshisanwo A.O., Dasiewicz P.P., "A Parallel Model and Architecture for Production Systems", *Proceedings of the 1987 International Conference on Parallel Processing*, pp. 147-153, 1987.

[19] Schreiner F., Zimmermann G., "PESA 1 - A Parallel Architecture for Production Systems", *Proceedings of the 1987 International Conference on Parallel Processing*, pp. 166-169, 1987.

[20] Ramnarayan R., Zimmermann G., Krolikoski S., "PESA-1: A Parallel Architecture for OPS5 Production System", *Proceedings of the Nineteenth Annual Hawaii International Conference on System Sciences*, pp. 201-205, 1986.

[21] Stolfo S.J., Miranker D.P., "DADO: A Parallel Processor for Expert Systems", *Proceedings of the 1984 International Conference on Parallel Processing*, pp. 74-82, 1984.

[22] Gupta A., "Implementing OPS5 Production Systems on DADO", *Proceedings of the 1984 International Conference on Parallel Processing*, pp. 83-91, 1984.

[23] Shaw D.E., "NON-VON: A Parallel Machine Architecture for Knowledge-Based Information Processing", *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 961-963, 1981.

[24] Gupta A., Forgy C.L., Kalp D., Newell A., Tambe M., "Parallel OPS5 on the Encore Multimax," *Proceedings of the 1988 International Conference on Parallel Processing, Vol. 1*, pp. 271-280, 1988.

[25] Sartori M.A., Antsaklis P.J., "Neural Computing and Production Systems," *Proceedings of the Third Annual Symposium on Intelligent Control*, Aug. 1988.

[26] Passino K.M., Sartori M.A., Antsaklis P.J., "Neural Computing for Numeric/Symbolic Conversion in Control Systems," *IEEE Control Systems Magazine*, pp. 44-52, April 1989.

[27] Passino K.M., Sartori M.A., Antsaklis P.J., "Neural Computing for Information Extraction in Control Systems," *Proceedings of the Twenty-Sixth Annual Allerton Conference on Communications, Control, and Computing*, University of Illinois at Urbana-Champaign, pp. 1172-1180, September 1988.

[28] Lippmann R.P., "An Introduction to Computing with Neural Networks," *IEEE ASSP Magazine*, pp.4-22, April 1987.

[29] McDermott J., Forgy C., "Production System Conflict Resolution Strategies," D.A. Waterman, F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

[30] Hopfield J., "Neurons with Graded Response have Collective Computational Properties like Those of Two-State Neurons", *Proceedings of the National Academy of Science*, pp. 3088-3092, May 1984.

[31] Kohonen T., *Self-Organization and Associative Memory*, Springer-Verlag, New York, NY, 2nd edition, 1988.

[32] Rumelhart D.E., Hinton G.E., Williams R.J., "Learning Internal Representations by Error Propagation", in Rumelhart D.E., McClelland J.L. (Eds.), *Parallel Distributed Processing: Explorations in the MicroStructrure of Cognition. Vol. 1: Foundations*, MIT Press, Cambridge, MA, 1986.

[33] Kohavi Z., *Switching and Finite Automata Theory*, McGraw-Hill, New York, NY, 1978.

[34] Sartori M.A., Passino K.P., Antsaklis P.J., "An Artificial Neural Network Solution to the Match Phase Problem in Rule-Based Artificial Intelligence Systems," Technical Report #89-09-02, Department of Electrical and Computer Engineering, University of Notre Dame, September 1989.

ANALYSIS OF BACK PROPAGATION ALGORITHM FOR FEED-FORWARD NEURAL NETWORKS

HENG GUO AND SAUL B. GELFAND
School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907

In this paper we summarize some properties of gradient-type learning algorithms as they apply to multilayer feed forward neural networks. The popular Back Propagation (BP) algorithm is such an algorithm which minimizes a mean-square error cost function [1].

We assume that the input pattern $\mathbf{x}$ and the desired output $\mathbf{y}$ are random variables. Define the cost $J(\mathbf{w}) = 1/2\ E(\|\mathbf{y}-\hat{\mathbf{y}}(\mathbf{w},\mathbf{x})\|^2)$, where $\hat{\mathbf{y}}$ is the network output and $\mathbf{w}$ is a vector containing the weights of the neural network. A gradient descent algorithm for minimizing J is

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \nabla J(\mathbf{w}_k), \tag{1}$$

where μ is a positive step-size parameter. In BP the gradient of J is replaced by a noisy or instantaneous gradient. Therefore BP is a stochastic gradient algorithm to which (1) represents the nominal response [2]. As a first step towards a rigorous analysis of BP we consider (1), which adjusts the weights based on known joint statistics of the input $\mathbf{x}$ and the desired output $\mathbf{y}$. We also assume a feed-forward network with a single hidden layer and linear threshold functions. Thus the input-output mapping of the network is given by $\hat{\mathbf{y}}=\mathbf{ABx}$, where $\mathbf{A}$ and $\mathbf{B}$ are matrices containing the weights of the output layer and the hidden layer respectively. Our goal is to demonstrate certain interesting dynamical properties of BP which hold for both linear and nonlinear threshold functions. We validate the analysis by using a training sequence with both linear and nonlinear threshold functions.

Our analysis and numerical work show that (1) exhibits some unusual behavior which arises from the fact that the set of locally optimal weights is highly degenerate. Indeed, even if there are no strictly locally optimal weights, the set of globally optimal weights forms a smooth, unbounded manifold in the weight space. Consequently, (1) behaves like a neutral system, i.e. any perturbation of the weight trajectory will permanently alter the trajectory and the perturbed trajectory, though staying near the original trajectory, will converge to a different point. Furthermore, a stochastic gradient algorithm (based on a training sequence) will have weight trajectories which chatter around the optimal manifold and may in fact drift along the manifold. The analysis is outlined below. First we study the continuous system

$$\dot{\mathbf{w}} = -\nabla J(\mathbf{w}). \tag{2}$$

(2) usually provides us with a good idea about the behavior of the solutions of (1) when μ is small [3]. We then analyze (1) by linearizing it along a solution of (2). Under our assumptions (2) can be put in the following form

$$\frac{d\mathbf{A}}{dt} = -(\mathbf{ABR}-\mathbf{P}^T)\mathbf{B}^T, \quad \frac{d\mathbf{B}}{dt} = -\mathbf{A}^T(\mathbf{ABR}-\mathbf{P}^T) \tag{3}$$

where $\mathbf{R}=E(\mathbf{x}\mathbf{x}^T)$, $\mathbf{P}=E(\mathbf{x}\mathbf{y}^T)$. Note that the set of optimal weights $E' = \{\mathbf{w}\text{: } \mathbf{ABR}-\mathbf{P}^T=0\}$ is an unbounded manifold in the weight space. Using LaSalle's theorem it can be shown [4] that the solutions of (3) are bounded and converge to the critical set $E = \{\mathbf{w}\text{:}\nabla J(\mathbf{w})=0\}$, which contains E' as a proper subset. It follows that for any $\epsilon>0$, there exists $\mu_0>0$ such that $d(\mathbf{w}_k,E)<\epsilon$ for $0<\mu<\mu_0$ and some finite k [3], where $d(\cdot,\cdot)$ denotes the Euclidean distance. Linearizing (1) along a trajectory $\mathbf{w}^*$ of (2) and assuming $\mathbf{w}^*$ is close to the optimal manifold yields

$$\triangle\mathbf{w}_{k+1} = (\mathbf{I} - \mu\mathbf{F}(\mathbf{w}_k^*)\mathbf{F}^T(\mathbf{w}_k^*))\triangle\mathbf{w}_k \tag{4}$$

where $\triangle\mathbf{w}_k=\mathbf{w}_k-\mathbf{w}_k^*$, $\mathbf{w}_k^*=\mathbf{w}^*(\mu k)$, and $\mathbf{F}$ is a matrix with fewer columns than rows. Thus the semi-definiteness of $\mathbf{F}(\mathbf{w}_k^*)\mathbf{F}^T(\mathbf{w}_k^*)$ indicates that locally (3) behaves like a neutral dynamical system. As a result $\mathbf{w}_k$ may drift along the optimal manifold under persistent perturbations, as is the case when a stochastic gradient is used. It can be shown [4] that a sufficient condition on the step size μ for the stability of (4) is given by

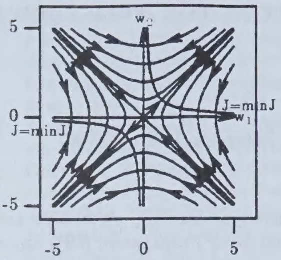


Fig.1 Optimal manifold and continuous Descent Paths of the linear net

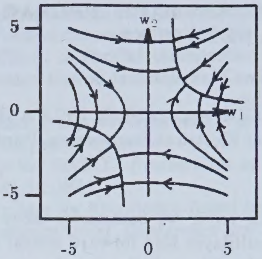


Fig.2 Optimal manifold and continuous descent paths of the nonlinear net

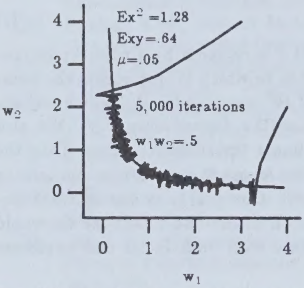


Fig.3 Weight drifting of the stochastic gradient algorithm in the linear threshold function case

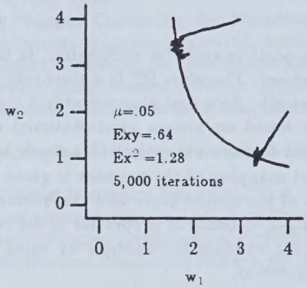


Fig.4 Weight drifting of the stochastic gradient algorithm in the nonlinear threshold function case

$$0<\mu<\frac{2}{\|\mathbf{R}\|\sup_{k}(\|\mathbf{A}_k^*\|^2+\|\mathbf{B}_k^*\|^2)}. \tag{5}$$

Numerical results for a simple two-layer-two-weight neural network are presented in Fig.1 through Fig.4. The network mapping is given by $\hat{y}=f(w_1 f(w_2 x))$, where $f(\bullet)$ is the threshold function. Both linear and sigmoid nonlinear threshold functions are used for comparison. The training sample is generated as following: $y=\pm.8$ with equal probability, $x=y+\nu$, where ν is Gaussian with mean zero and variance σ^2, and y and ν are independent. In the linear case the true gradient can be found in terms of the second order statistics $E(xy)$ and $E(x^2)$, and the weights **w**. The trajectories and the set of optimal weights can be derived analytically and are shown in Fig.1. In the sigmoid nonlinear case the true gradient is approximated by averaging over a set of 1,000 samples, and the trajectories and the set of optimal weights were computed numerically and are shown in Fig.2. Figures 3 and 4 show the weight drifting in both cases. These results show the similar dynamical properties between the gradient descent algorithms in the two cases.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Internal Representations by Error Propagation," in "Parallel Distributed Processing," Vol. 1. D. E. Rumelhart & J. L. McClelland (Edt), The M.I.T. Press, Cambridge Massachusetts, 1986.

[2] S. Haykin, "Adaptive Filtering Theory," Prentice-Hall, New York, 1985.

[3] D. Russell, "Optimization Theory," W. A. Benjamin, Inc., New York,1970.

[4] H. Guo and S. B. Gelfand, "Analysis of Back Propagation Algorithm for Multilayer Neural Networks", in preparation.

PERIODIC AND PULSE FUNCTION GENERATION USING A NEURAL NETWORK*

D. L. Gray and A. N. Michel
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, Indiana 46556

SUMMARY

In the present paper we describe a system with an imbedded feedback neural network which addresses the following problem: given a signal which is representable by a piecewise continuous function $f: [t_0, t_0 + T] \rightarrow R$, $t_0 \geq 0$, $T > 0$, we wish to generate a signal $\tilde{f}: [t_0, t_0 + T] \rightarrow R$ such that for a specified $\varepsilon > 0$, $| f(t) - \tilde{f}(t) | < \varepsilon$ for all $t \in [t_0, t_0 + T]$.

The generation of arbitrary analog waveforms is presently accomplished by utilizing a combination of a digital system (CPU) and a digital to analog converter. In this approach a desired sampled waveform is generated from a difference equation, look up table, or the like. This digital approximation is then processed by a D/A converter to yield the desired (approximate) analog signal ([1], [2], [3]). Methods of this type utilize a uniform spacing between samples, as governed by the sampling rate. Furthermore, the maximum speed at which the system can run is restricted by the rate at which sampled data can be generated and/or recalled, transferred to and reconstructed by the digital to analog converter.

The proposed system consists of three main parts: a ramp generator, a feedback neural network, and an analog to digital converter. We propose to utilize a feedback neural network with lower block triangular form to map the ramp function into a digital representation of the approximation of time function f. The digital representation is then reconstructed via a D/A converter to produce the output function $\tilde{f}$.

---

*The work of D.L. Gray and A.N. Michel is supported by the National Science Foundation under grant ECS 88-02924.

The method utilized to design the feedback neural network with lower block triangular form was discussed in [4],[5], and [6]. This procedure will create the bias and interconnection terms of a feedback neural network from a data table. The data table determines a partition of the time interval $S = [\, t_0, t_0 + T \,]$ given by $S_i \subset S$ i=1,...,k such that $S_i \cap S_j = \emptyset$ $i \neq j$, and $\cup_{i=1}^{k} S_i = S$. Associated with each subinterval S_i is a desired digital output word Y of length M representing the approximation $\tilde{f}$ of the function f to which each subinterval S_i corresponds.

Our method has several advantages over existing (digital) techniques of generating test signals ([1], [2], [3]) including:

a) the removal of the digital cpu unit and memory;
b) the ability to use non-uniform sample spacing to recreate the waveform;
c) the potential of generating waveforms higher in frequency than is obtainable by the current methods ([1], [2], [3].).

References

[1] D. F. Hoeschele, Jr., **Analog-to-Digital/ Digital-to-Analog Conversion Techniques,** John Wiley and Sons, 1968.

[2] Precision Digital Sine-Wave Generation with the TMS32010, Texas Instruments, 1984.

[3] TMS 32010 Digital Signal Processor Design Workshop RTC-110, Texas Instruments, 1985.

[4] D.L. Gray, A. N. Michel, and W. Porod, "Application of Neural Networks to Sorting Problems", Proceedings of the 27th IEEE Conference on Decision and Control, Austin, Texas, Dec. 1988, p350-351.

[5] D.L. Gray, A. N. Michel, and W. Porod, "Neural Networks with Lower Block Triangular Interconnecting Structure", Proceedings of the Com Con-88: Advances in Comunication and Control Systems, Baton Rouge, Louisiana, Oct. 1988, p688-699.

[6] A. N. Michel and D. L. Gray, "Analysis and Synthesis of Neural Networks with Lower Block Triangular Form", submitted to IEEE Transactions on Circuits and Systems.

An Efficient Neural Network Realization of Nonlinear Functions*

Qiu Huang and Ruey-wen Liu
Department of Electrical and Computer Engineering
University of Notre Dame
Notre Dame, IN.46556

Abstract

An efficient constructive method is presented in this paper to realize any continuous nonlinear functions with a neural network. This neural network consists of only one hidden layer. The number of nodes in hidden layer can be estimated. A particular initial condition is chosen so that the backpropagation algorithm converges to a solution with a error between the neural network realization and the given function less than a given $\delta > 0$*. A circuit for nonlinear function fitting built with a neural network has an advantage over the tradition circuits. It is more realiable, easy adjustable, more accurate and can be designed and manufactured in a more convenient and regular way.*

I Introduction

The celebrated Hilbert's 13-th problem has stired up the interest among mathematicians a more general question: When can a complex function be represented by a combination of finite number of simple functions ?

In 1957, Kolmogorov [1] published an astounding theorem:

Theorem 1 : There exists a finite number of continuous increasing functions $h_{ij}(x)$, on $I=[0,1]$ so that each continuous function f on I^n can be written in the form :

$$f(x)=\sum_{j=1}^{2n+1} g_j\left(\sum_{i=1}^{n} h_{ij}(x_i)\right) \tag{1}$$

where g_j are properly chosen continuous function of one variable.

Hence, every continuous function of n variables can be represented by a (nonlinear) combination of continuous functions of a single variable. Note that the set $\{h_{ij}\}$ is finite and it does not depend on f.

---

* This work is supported by the Center for Signal and Imaging Processing with Medical Applications in Notre Dame.

Later G.G.Lorentz [2] improved the theorem by shown that the set of functions $\{g_j\}$ can be reduced to a single function g and D.A.Sprecher [3] has shown that the set of functions $\{ h_{ij} \}$ can be reduced to $\{ \mu_i h_j \}$ where μ_i 's are fixed constants. Hence, Equation (1) becomes

$$f(x)=\sum_{j=1}^{2n+1} g_j(\sum_{i=1}^{n} \mu_i h_j(x_i)) \tag{2}$$

However, there is no constructive methods to compute the functions g, $\{ h_j \}$ and the constants $\{\mu_i\}$.

Clearly, Equation (2) has an implication of a three-layer perceptron neural network representation. However, since the functions g and $\{ h_i \}$ are not known, one can not perform neuro-computation.

Recently, Cybenko [4] gave an approximation representation of f :

Theorem 2 : Let σ be any continuous sigmoidal function. Then the set of finite sums of the form

$$g(x)=\sum_{j=1}^{N} \alpha_j \sigma(\omega_j^T x+\theta_j) \tag{3}$$

are dense in $C(I^n)$, where α_j, $\theta_j \in R$, and $\omega_j \in R^n$.

Hence, any continuous function f in I^n can be approximated by a function g in the form of (3), as close as one wants. Again, there is no constructive method for the determination of N, as well as the constants α_i, θ_j and ω_j.

Note that because of its simple form, Equation (3) has a three-layer perceptron neuron networks realization as shown in Figure 1. The single nonlinear function σ can be chosen among sigmoidal functions. It remains to determine the constants α_j, ω_j, θ_j and N. We propose to do so by neuron computing. It may be important to note that neurocomputation is the *only* viable method available at the present time for the determination of these constants. Because of this, it brings out an unique capability of neural networks.

Lippman [5] has had his experience with neuro-computation. However, since there are hidden layers in the neural networks, currently, no existing algorithm can guarantee it to converge to a global minimum, i.e., a solution. Sometimes, it converges to a local minimum instead. In this paper, we propose a method which allows us to chose an initial condition so that it will converge to a solution g which can be as close to f as one wants.

II. The Main Result

First of all, we will chose the function

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{4}$$

to be the sigmoidal function in (3). As in [5], we will consider functions of single variable. As it was discussed, Equation (3) has a perceptron neuron network realization and the success of neuro-computation depends on the choice of its initial conditions. The correct choice of initial condition and number of nodes of hidden layer is based on the following theorem.

Theorem 3 : There exists a cannonical single variable function f(z), namely,

$$f(z)=1+|z+\frac{1}{2}|-|z-\frac{1}{2}| \tag{5}$$

such that the following is true: let h be any continuous function of the form

$$h(x) = \begin{cases} f(x) & x\in [x_0,x_m] \\ f(x_0) & x\in (-\infty,x_0] \\ f(x_m) & x\in [x_m, \infty) \end{cases} \tag{6}$$

where $[x_0,x_m]$ is a compact set. Then, given any $\delta > 0$, there exists a finite sum of the form

$$g(x)=\sum_{j=1}^{N}\alpha_j\phi(\omega_j x+\theta_j)+c \tag{7}$$

such that $\| h(x) - g(x) \|_\infty < \delta, \quad x\in R$

where $c = f(x_0)$.

Proof : It is clear that any continuous function can be represented by a piecewise-linear function as close as possible. Let g(x) be a piecewise-linear function such that

$$\| h - g \|_\infty < \delta \tag{8}$$

and $g(x) = f(x_0), \ x\in (-\infty,x_0]; \ g(x) = f(x_m), \ x\in [x_m, \infty)$. Next step , we need to prove that g(x) has a form as (7).

Assume that there are N+1 breakpoints, say , $x_0, x_1, x_2, \ldots, x_N=x_m$. We will prove it in an induction way. When N=1, it is clear that

$$g(x) = f(x_0) + \alpha\, \phi(\omega x + \theta)$$

where $\alpha = f(x_m) - f(x_0)$, $\omega = 1/(x_m - x_0)$, $\theta = (x_0 + x_m)/2(x_0 - x_m)$, which is a form of (7). Now assum that when N=k, g(x) has the form of (7), we need to prove that when N=k+1, g(x) has a form of (7). When N=k+1, there are k+2 breakpoints and k+3 segments including segments in $x \in (-\infty, x_0]$ and $x \in [x_m, \infty)$. Considering the first k+1 segments, we define a function

$$L(x) = \begin{cases} g(x), & x \in (-\infty, x_k) \\ g(x_k), & x \in [x_k, \infty); \end{cases}$$

Hence, the function L(x) can be represented in a form of (7).

Since

$$g(x) = L(x) + \alpha_{k+1}\, \phi(\omega_{k+1} x + \theta_{k+1})$$

with $\alpha_{k+1} = g(x_{k+1}) - g(x_k)$, $\omega_{k+1} = 1/(x_{k+1} - x_k)$,

$\theta_{k+1} = (x_k + x_{k+1})/2(x_k - x_{k+1})$,

g(x) can be represented in a form of (7) . Q.E.D.

Remark:

1) (7) will be satisfied for all $x \in R$, which is not compact, hence it is not a direct result of the Theorem 2, nor it can be proved by using the method in [4].

2) The function (5) itself is a sigmoidal function, but from a practical point of view, such a function cannot be built , and the close form of sigmoidal function in circuits is neither necessary the same as (4), nor the same with each other. Therefore, a neuroncomputing processing is need to find the coefficients

Our neural network will consists of one input layer, one hidden layer and one output layer as shown in Figure 1. Linear nodes are used for input and output layer. In hidden layer, the sigmoidal nonlinearity nodes (4) are used. Number of hidden nodes are determined by N in (7). Then a backpropagation algorithm is used to train this neural network with the data of f(x) sampled in $[x_0, x_m]$. The backpropagation algorithm is a modified gradient decent method. This algorithm suffers from converging to a local minimun instead of a global minimum. Hence, if we choose initial data for all parameters in a neural network arbitrary , there will be no guarantee to converge to a global minimum. Here by using the result of theorem 3, we first find an approximation of f(x) with g(x) in (7) however crude it may be. When g(x) is found , the corresponding breakpoints also given, say , x_0, x_1, x_2, …, $x_N=x_m$. Then the initial weighs in Figure 1 can be computed as follows:

$$\beta_i = f(x_{i+1}) - f(x_i)$$

$$\omega_i = \frac{1}{x_{i+1} - x_i}$$

$$\theta_i = \frac{x_{i+1} + x_i}{2(x_{i+1} - x_i)} \tag{9}$$

The key point is that those initial data can be computed simply and directly from the data of breakpoints and the corresponding value of f(x). We have proposed a method to determine the number of nodes in hidden layer, and how to choose initial weighs so that a good approximate solution can be achieved. We may not say the number of nodes in hidden layer N is minimum number for every cases, but this number is still small, and we believe that it is not far away from the minimum. We may also not say the initial condition chosen in this way always good for the algorithm to converge to a global minimum, but the following simulation results will show that this initial condition is good enough for the algorithm to converge to a good solution, in the sense that the relative error between the actual nonlinear function f(x) and the function $\bar{f}(x)$ of a neural network is less than 6%, with a certain iterate steps. An extensive computer simulation study has demonstrated our method.

III. Simulation

In this section, two examples will be used to demonstrate our method.

Example 1 Consider a continuous nonlinear function

$$f(x) = 0.5e^{-x^2} - 0.8e^{-(x-4)^2} + e^{-0.5(x+4)^2} \tag{10}$$

we will realize it by a neural network in Figure 1 in the interval [-10,10]. The N=7 breakpoints are chosen as

$$[-6, -4, -1.5, 0, 2, 4, 6]$$

The initial data for β_i , ω_i and θ_i can be computed by (9),

$$\beta = [1.0,\ -0.9,\ 0.4,\ -0.5, -0.8, 0.8]$$

$$\omega = [0.5, 0.4, 0.6667, 0.5, 0.5, 0.5]$$

$$\theta = [-2.5, -1.2, -0.6667, 0.5, 1.5, 2.5]$$

Then a neural network with only 6 nodes in hidden layer is trained by using a backpropagation algorithm with step size 0.1. The data used for training purpose are sampled in [-10,10] with sampling rate 1/100. For the sake of comparison, we also try a two-hidden-layer neural network suggested in [5], with 20 nodes in the first hidden layer, and 5 nodes in the second layer. Simulation results show that by using our method, when k=5000 steps , the relative error ,both norm-2 and norm-infinite are below 10%, as shown in Figure 2. After 10^4 steps, the relative error is no more than 7%, and the final parameters of a neural network become

$$\beta = [1.5579, -1.7604, 0.9089, -0.6416, -1.7616, 1.7264]$$

$$\omega = [0.5638, 0.5320, 0.8565, 0.7919, 0.6845, 0.7026]$$

$$\theta = [-2.7178, -1.5711, -0.7347, 0.3823, 2.3927, 3.1694] .$$

But the method used in [5] cannot fit f(x) even after $5\text{x}10^4$ iteration steps (actually, after 5000 steps, the error never decrease) as shown in Figure 3. In Figures 2 and 3, the dotted line curve represents f(x) in (10), the solid line represents $\bar{f}(x)$, i.e. the function of a neural network.

Example 2 In this example, we try to use a neural network to build a 'period' signal in the interval [-10,10].

$$f(x) = 0.5 \sin(x) + 0.25 \tag{11}$$

The N=8 breakpoints are chosen as

$$[-10, -8, -4, -2, 2, 4, 8, 10]$$

After computing the initial data by using (9), a neural network with only 7 nodes in hidden layer is trained with same sampling rate as in example 1. After 10^4 steps the relative error of both norm 2 and norm infinite is below 6%, as shown in Figure 4. The parameters of a neural network become

$$\beta = [-1.0608, 1.5415, -1.4513, 1.4104, -1.4204, 1.3107, -0.9801]$$

$$\omega = [0.7315, 0.4680, 0.4862, 0.5872, 0.5165, 0.5577, 0.7288]$$

$$\theta = [-6.5778, -2.9799, -1.5424, -0.0042, 1.6143, 3.4029, 6.7518]$$

Remark:

1) In both examples above , the chosen-breakpoints are crude, therefore the piecewise linear curve g(x) does not fit f(x) very well . But with these data as an initial codition , the $\bar{f}(x)$ of neural network is goingto fit f(x) when the relative error become smaller .

2) The relative error of norm-2 and norm-infinite is close to each other, after several iteration

steps, which implies that the curve is fitting smoothly and without any spike.

3) The number of nodes in hidden layer is no more than 7. More importantly, we have the good estimation for N, insteadof random tries.

IV Application

Here, we give a simple application of this method to the design of a negtive device. We use the simple transconductance amplifier [6] as the node in hidden layer as shown in Figure 5. The procedure for the design of a negative device by using a neural network is first to determine the number of nodes in hidden layer of a neural network, say N, then to simulate this neural network to obtain all parameters of a neural network. With these parameters and N simple transconductance amplifiers, the negative device can be built.

Consider the driving-point v-i characteristics of a negative device denoted as dotted line in Figure 6, only three breakpoints are chosen as

$$[0, 0.3, 1]$$

We use only two simple transconductance amplifiers to build these negative devices. After a simulation as shown in Figure 6, the parameters can be obtained as

$$\beta = [3.05, \ -3.09]$$

$$\omega = [4.70, 2.03]$$

$$\theta = [0.75, 0.71]$$

The relative error between actual characteristic and the output curve of a neural network is less than 4%.

If the characteristic denoted as dotted line in Figure 7 has a narrow pulse. We use three nodes with a chosen-breakpoints as [0, 0.05, 0.2, 0.4]. After a simulation as shown in Figure 7, the parameters become

$$\beta = [6.60, -4.23, -2.40]$$

$$\omega = [20.95, 6.28, 5.31]$$

$$\theta = [-0.07, -0.40, 0.52]$$

The relative error between actual characteristic and ones of a neural network is less than 5%.

We can use the method described above to build circuits fitting the nonlinear v-i characteristics of a device. The circuits for nonlinear function fitting built in a neural network has an advantage over a tradition method . It is more reliable, more accurate, easy adjustable. More imporetantly, it can be designed and manufactured in a regular way as shown in a application to the design of a negative device above.

References:

[1] A.N.Komogorov " On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of One Variable and Addition", Dokl. Akad. Nawk SSSR 114, 1957 (pp.953-956) English translation, American Math. Soc. Transl. (2), 28(1963), pp.55-59, MR. 22#2669; 27#3760.

[2] G.G.Lorentz " The 13-th Problem of Hilbert". In F.E.Browder editor, Mathematical Developments Arising from Hilbert Problems, American Mathematical Society, Providence, R.I., 1976.

[3] A.Lapedes and R. Farber " How Neural Nets Work". In D.Anderson, editor, Neural Information Processing Systems, pages 442-456, New York, 1988. American Institute of Physics.

[4] G.Cybenko " Approximations by Superpositions of A Sigmoidal Function" Mathematics of Control, Signals and System, Vol.2, pp.303-314, 1989.

[5] R.P.Lippmann and P.Beckman " Adaptive Neural Net preprocessing for Signal Detection in Non-guassian Noise", In David S. Towetzky editor, Advance in Neural Information Processing Systems I, pages 124-132, Denver, 1988.

[6] C. Mead Analog VLSI and Neural Systems , Addison-wesley Publishing Company. 1989.

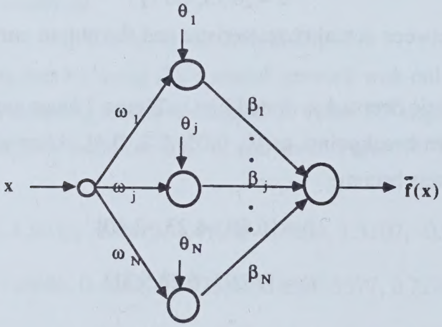
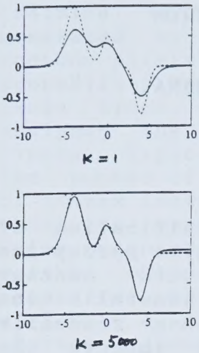


Figure 1 A Neural Network



K = 1 K = 1000

K = 5000 K = 10^4

Figure 2 Simulation Results in Example 1 with our method

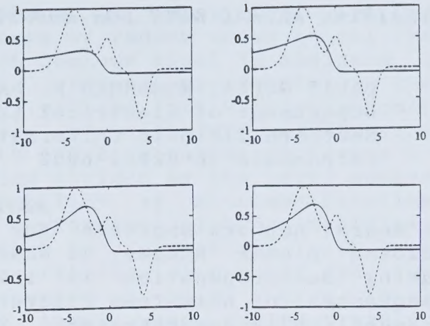
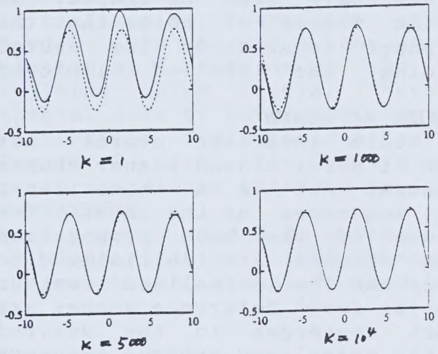


Figure 3 Simulation Results in Example 1 with the method in [5]



K = 1 K = 1000

K = 5000 K = 10^4

Figure 4 Simulation Results in Example 2 with our method

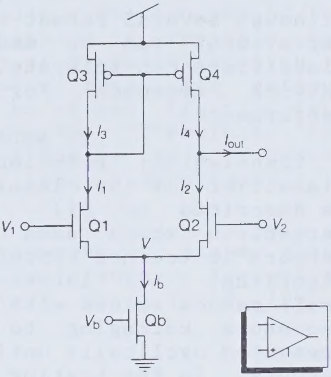
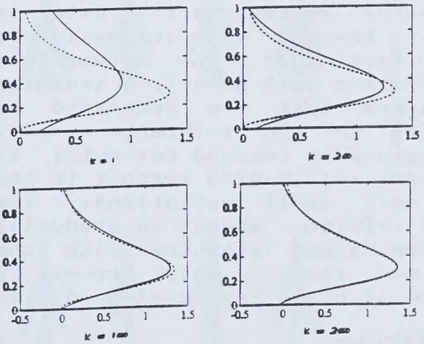


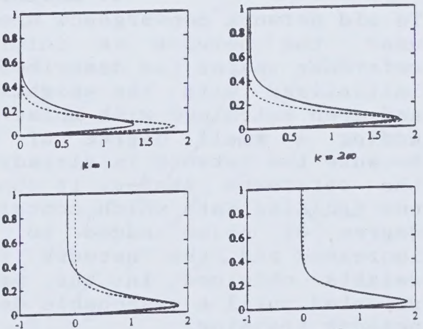
Figure 5 A Simple Transcondutance Amplifier



k = 1 k = 200

k = 1000 k = 2000

Figure 6 Curve Fitting For The Characteristic Of A Negtive Device



k = 1 k = 200

k = 1000 k = 2000

Figure 7 Curve Fitting For The Characteristic Of A Negtive Device With Narrow Pulse

TRAINING NEURAL NETS FOR ROBUST SHAPE CLASSIFICATION

LALIT GUPTA, MOHAMMAD R. SAYEH, AND RAVI TAMMANA
Department of Electrical Engineering
Southern Illinois University at Carbondale
Carbondale IL 62901-6603

ABSTRACT

A neural network approach for the robust classification of closed planar shapes is described. A three layer perceptron using backpropogation is initially trained with contour sequences of noisefree reference shapes and its generalization capability is demonstrated. The network is then gradually retrained with increasingly noisy data to improve the robustness of the classifier. The advantages and improvement in robustness using this extended training scheme are shown and typical classification results are presented.

1. INTRODUCTION

Although several robust classifiers have been developed, an improvement can be made in the degree of noise that the classifiers can tolerate. This paper focuses on the neural network approach for providing the desired improved performance.

2. NEURAL NETWORK APPROACH

A translation, rotation, and scale invariant neural net classifier for the classification of noisy closed planar shapes is described in [1]. This neural net is a three layer perceptron which uses contour sequences as the input. The network is trained through the use of the back propogation algorithm. All interconnection weights are initialized to small random values with zero mean and the normalized contour sequences belonging to the "noise free" reference shapes are presented cyclically until the net converges to the desired outputs. In the testing stage, the normalized contour sequence of the test shape is presented to the network and the test shape is assigned to the class of the output that yields the maximum value.

3. EXTENDED TRAINING

To aid network convergence when noisy representative data is used, the network is initially trained with the noisefree reference shapes (as described in Section 2). The network is initialized with the weights obtained with noisefree training and then retrained with noisy shapes that are generated by adding a small degree of noise to the reference shapes. Because the network is already trained to respond correctly to the reference shapes, it converges easily with respect to the new training data which contains only small variations. The degree of noise added to the reference shapes is gradually increased and the network is retrained starting with the weights obtained in the previous stage. This process is repeated until a reasonable degree of noise has been used for network training.

4. SIMULATIONS

The performance of the proposed neural net classifier is demonstrated by considering the classification of shapes belonging to four classes. The reference shapes are shown in Figure 1. A series of tests were conducted to obtain classification results for both noisefree training and extended training. From each reference shape, a noisy test set is generated by adding various degrees of random noise to all the contour pixels of the shape. Each contour pixel is assigned a probability p of retaining its original coordinates in the image plane and a probability $q=(1-p)$ of being randomly assigned the coordinates of one of its eight neighbouring pixels. Typical noisy shapes with $q=0.5$ are shown in Figure 2. The number of shapes misclassified divided by the total number of shapes tested gives the probability of misclassification which is used as a measure of performance of the classifier. Using extended training, the network is initialized with the weights obtained with noisefree training. Ten noisy training shapes per class were generated using a small noise level $q_t=0.1$. q_t denotes the noise level used during training whereas q denotes the degree of noise in the test shapes. The network is retrained with the 40 low noise level shapes and the performance is evaluated for test shapes with various values of q. The network is then initialized with the weights obtained with $q_t=0.1$, retrained with 40 shapes generated with $q_t=0.2$, and re-tested with different q values. The process of re-initialization, retraining, and re-testing is repeated for $q_t=0.3$. In Figure 3, the probability of misclassification is plotted as a function of the test noise level for various training noise levels. Note that noisefree training is a special case of extended training with $q_t=0$. Therefore, the performance for noisefree training is given by the $q_t=0$ curve. Figure 3 also clearly demonstrates the improvement in performance with extended training.

REFERENCES

1. L. Gupta and M.R. Sayeh, "Neural networks for planar shape classification," IEEE International Conference on Acoustics, Speech, and Signal Processing, New York, April 11-14, 1988.

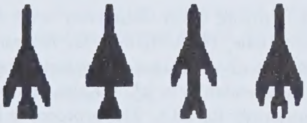


Figure 1

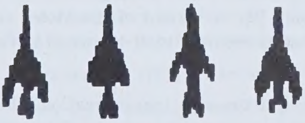


Figure 2

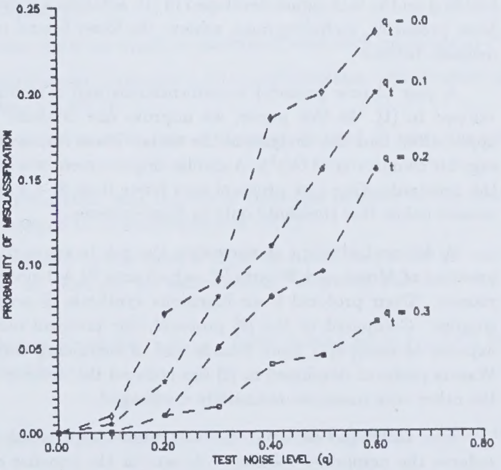


Figure 3

A BYZANTINE AGREEMENT PROTOCOL WITH OPTIMAL MESSAGE BIT COMPLEXITY

BRIAN A. COAN
Bell Communications Research
Morristown, NJ 07960–1910

JENNIFER L. WELCH
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract: This paper presents a new Byzantine agreement protocol that tolerates t processor faults using $3t+1$ processors, $t+o(t)$ rounds, $O(t^2)$ total message bits, and local computation polynomial in t. The protocol is optimal or near optimal in all cost measures: the number of processors is optimal, the message bit complexity is optimal, and the number of rounds exceeds the lower bound by $o(t)$. This is the first Byzantine agreement protocol to have optimal message bit complexity.

1. INTRODUCTION

Byzantine agreement is a fundamental problem, first introduced by Pease, Shostak, and Lamport [10]. Many variations of this problem have been studied — we consider deterministic protocols for synchronous systems subject to Byzantine processor faults. In this model, there is currently a gap between the known lower bounds on resources and the most efficient known protocols. The results in this paper help narrow that gap.

There are four cost measures that interest us: number of processors, rounds of communication, total number of message bits, and amount of local computation. If t is an upper bound on the number of processor faults, then the following lower bounds are known: $3t+1$ processors [10, 7], $t+1$ rounds [6], $\Omega(t^2)$ message bits [4], and local computation polynomial in t. The lower bounds on processors and rounds were known to be achievable. For example, the original protocol in [8] achieves these bounds (but requires a number of message bits that is exponential in t). The best previous bound on message complexity is in the recent protocol in [1], requiring $O(t^{2+\epsilon})$ total message bits. Our new protocol, building on the techniques developed in [1], achieves a message bit complexity of $O(t^2)$, which is optimal. Most protocols, including ours, achieve the lower bound on local computation; we will not discuss this measure further.

A pair of new protocol transformations and a technique for applying them recursively were developed in [1]. In this paper, we improve one of these transformations, the technique for recursive application, and the analysis of the costs. These improvements enable us to achieve the optimal message bit complexity of $O(t^2)$. A similar improvement was reported independently in [2]. We also improve the constants. Our new protocol uses fewer than $2t+2$ rounds uniformly for all t. The protocol in [1] crosses below this threshold only in huge systems.

A different attempt at narrowing the gap in resource requirements for Byzantine agreement is the protocol of Moses and Waarts [9], which uses $6t+1$ processors, $O(t^8)$ message bits, and exactly $t+1$ rounds. Their protocol is an ingenious synthesis of several complicated techniques; our approach is simpler. Compared to the [9] protocol, our protocol uses fewer processors and message bits, at the expense of using $o(t)$ more rounds and of sacrificing early stopping [5]. A variant of the Moses and Waarts protocol developed in [2] has reduced the number of processors required to $4t+1$ while leaving the other cost measures essentially unchanged.

Our technique consists of the recursive application of a general two-step transformation which reduces the number of message bits sent at the expense of a small increase in the rounds. Our transformation leaves unchanged the number of processors and the number of faults tolerated. A single application of the transformation, at the expense of adding $o(t)$ rounds of communication, greatly reduces the number of message bits sent.

The two steps of our transformation are as follows. First, we transform any Byzantine agreement protocol (that satisfies certain technical conditions) into a protocol for the Byzantine broadcast problem. In a Byzantine broadcast protocol, a small committee (subset of the processors) attempts to reach agreement and to communicate the decision to all the processors. The first step of our transformation yields a Byzantine broadcast protocol that operates correctly as long as there are enough nonfaulty processors in its committee. Second, we produce a Byzantine agreement protocol from a collection of Byzantine broadcast protocols that have disjoint committees. For the combined protocol to work correctly, it is sufficient that the fault-tolerance assumption of any one of its constituent Byzantine broadcast protocols be satisfied—it does not impair the correct operation of the combined protocol if the committees of the remaining Byzantine broadcast protocols have "too many" faulty processors. Thus, we can combine B Byzantine broadcast protocols, tolerating $t_1, \ldots, t_B$ faults respectively, to produce a Byzantine agreement protocol tolerant of $t_1 + \ldots + t_B + B - 1 = t$ faults overall. The resulting protocol satisfies the technical conditions referred to above, and thus our transformation can be applied recursively.

2. MODEL

We consider computer systems consisting of a collection of processors that interact by sending messages over links. Each link provides a bidirectional communication path between a pair of processors. The collection of links constitutes the communication network. It is known [3, 7] that no agreement protocol is possible unless the connectivity of the communication network is high. Throughout this paper we assume that the communication network is fully connected and reliable.

We assume a *synchronous* timing model, meaning that all processors take steps in unison and all messages are received a fixed amount of time after they are sent. An execution of a protocol consists of a series of *rounds.* In each round each processor sends messages (whose contents depend on its internal state), receives all the messages just sent to it, and acts on the messages received (by altering its internal state). The protocol prescribes what messages should be sent and how internal states should be altered.

In an execution of a protocol, some processors are nonfaulty, and the rest are faulty. A *nonfaulty* processor sends messages and changes state according to the protocol. A *faulty* processor can deviate from the protocol. We are interested in protocols that tolerate *Byzantine* processor faults. In the Byzantine fault model there are no constraints on the contents of messages sent by a faulty processor; however, the recipient of any message knows the identity of the sender.

Fix a set P of *processors* (actually, processor *names*). A *message* is any bit string, marked with the names of two processors, the *sender* and the *recipient.* (We do not charge for the sender's and recipient's names in calculating message complexity.)

A *protocol* for P is a set of state machines, one state machine associated with each processor in P.

- a set of *states*, containing a subset of *initial* states, partitioned into v-initial states, and a subset of *final* states, partitioned into v-final states, $v \in \{0, 1\}$.
- a *message generation function* from the current state to a set of messages to send, at most one per processor; and
- a *state generation function* from the current state and the set of messages received to the next state, such that v-final states are only mapped to v-final states, $v \in \{0, 1\}$.

A *round* is a triple $\langle S_1, M, S_2 \rangle$, where S_1 and S_2 are each a tuple of states, one state for each processor in P; and M is a set of messages, at most one for each (sender, recipient) pair in $P \times P$. Denote element S_1 of round r by $r(S_1)$, etc.

An *execution* is a finite or infinite sequence of rounds $r_1, r_2, \ldots$, such that $r_i(S_2) = r_{i+1}(S_1)$ for all i. Processor p in P is *nonfaulty* in an execution if

- p's state in $r_1(S_1)$ is an initial state;
- the set of messages sent by p in $r_i(M)$ is the result of p's message generation function applied to p's state in $r_i(S_1)$, for all i; and
- p's state in $r_i(S_2)$ is the result of p's state generation function applied to p's state in $r_i(S_1)$ and the set of messages in $r_i(M)$ sent to p, for all i.

Otherwise, p is *faulty* in the execution. Thus a faulty processor can make arbitrary (local) state changes, and send arbitrary messages (but at most one message to any processor in any round).

In an execution, if processor p starts in a v-initial state, then v is p's *input*; if p enters a v-final state, then v is p's *output*.

Our protocols will call other protocols as subroutines, using the following syntax and semantics. Let $\mathcal{P}$ and $\mathcal{P}'$ be protocols for P and P' respectively. Processor p in P, during an execution e of protocol $\mathcal{P}$, may execute

call $\mathcal{P}'$(rounds:R, input:x, output:y).

We require that P' be a subset of P, and that every processor in P' call $\mathcal{P}'$ in the same round of e as p, and with the same (constant) value of R. (This is a syntactic requirement on the code, not a requirement on the behavior of faulty processors.) The effect of the call is that p gets a new set of variables, as specified by the state machine for p in $\mathcal{P}'$, initialized to an x-initial state; p computes according to $\mathcal{P}'$ for R rounds; and p returns y, where p ends in a y-final state after the R rounds ($y = \perp$ if p is not in a final state).

Processors in our protocols manipulate elements from the set $\{0, 1, \perp\}$ (in addition to possibly other elements, depending on the subprotocols); 0 and 1 are *values*, but $\perp$ is not. Let *default* be an arbitrary fixed value. The number of bits needed to represent a value or $\perp$ is a constant, and thus will not be included in our complexity analyses.

The *processor complexity* of a protocol (for P) is $|P|$. The *round complexity* of a protocol is the maximum, over all executions e, of the minimum number of rounds of e required for all nonfaulty processors to be in final states. The *message bit complexity* of a protocol is the maximum, over all executions e, of the total number of bits sent in messages by nonfaulty processors in e. The *message size complexity* of a protocol is the maximum, over all executions e, of the number of bits sent in any single message by any nonfaulty processor in e. The *last-round senders complexity* of a protocol is the maximum, over all executions e, of the number of nonfaulty processors that send a message in the last round of e. (The last two of these complexity measures, message size and number of last-round senders, have been introduced to facilitate our analysis of the message bit complexity.)

3. PROBLEM STATEMENTS

This paper describes efficient protocols for the Byzantine agreement problem. Our protocols use as subroutines solutions to a closely related problem, which we call the Byzantine broadcast problem. We now define these two problems.

3.1 Byzantine Agreement

A protocol (for P) is a *Byzantine agreement protocol for P and t*, $t < |P|$, if there exists an integer $R \geq 0$ such that every execution at least R rounds long satisfies the following three conditions.

- *Termination:* At the end of R rounds every nonfaulty processor in P is in a final state.
- *Agreement:* If at most t processors in P are faulty, then there is a $v \in \{0, 1\}$ such that any nonfaulty processor in P that enters a final state enters a v-final state.
- *Validity:* If at most t processors in P are faulty and there is a $v \in \{0, 1\}$ such that all nonfaulty processors in P begin in a v-initial state, then any nonfaulty processor in P that enters a final state enters a v-final state.

We impose the nonstandard requirement that a nonfaulty processor must eventually enter a final state, even if more than t processors are faulty. The more standard termination requirement, used elsewhere in the literature, is that a nonfaulty processor must eventually enter a final state in any execution in which t or fewer processors are faulty. Most, but not all, protocols which satisfy the standard termination requirement can trivially be transformed into protocols that satisfy our termination requirement. In particular, this transformation is possible for any protocol for which there is a known upper bound, possibly a function of $|P|$ and t, on the number of rounds needed for termination.

By a result in [8], no Byzantine agreement protocol for P and t can exist unless $|P| > 3t$. We will concentrate on the situation $|P| = 3t + 1$. Since processors are in general an expensive resource, it is reasonable to minimize the total number of processors needed to achieve a certain level of fault-tolerance, as represented by t. However, our Byzantine agreement protocols will work if $|P| > 3t + 1$.

3.2 Byzantine Broadcast

The Byzantine broadcast problem is essentially the problem of having a known, fixed subset of the processors perform Byzantine agreement and communicate the result to all of the processors. Thus the subset of the processors causes all the nonfaulty processors to choose a value, in such a way that if enough members of the subset are nonfaulty, then (1) all the nonfaulty processors choose the same value, and (2) if all the members of the subset begin with the same value, then all the nonfaulty processors choose that value. The identity of each member of this subset is known to all processors.

A protocol (for P) is a *Byzantine broadcast protocol for* P, C, *and* t', $C \subseteq P$, $t' < |C|$, if there exists an integer $R \geq 0$ such that every execution at least R rounds long satisfies the following three conditions.

- *Termination:* At the end of R rounds every nonfaulty processor in P is in a final state.
- *Agreement:* If at most t' processors in C are faulty, then there is a $v \in \{0,1\}$ such that any nonfaulty processor in P that enters a final state enters a v-final state.
- *Validity:* If at most t' processors in C are faulty, and if there is a $v \in \{0,1\}$ such that all nonfaulty processors in C begin in a v-initial state, then any nonfaulty processor in P that enters a final state enters a v-final state.

3.3 Asymptotic Complexities

We have defined a Byzantine agreement protocol for a fixed P and t. There is a natural sense in which we would like to consider two Byzantine agreement protocols, for different sets of processors and different t's, to be instances of the same "protocol." We thus assume an infinite set of processor names U and define a Byzantine agreement *protocol family* to be an infinite set $\{\mathrm{BA}(n,t) : n \geq 1, t \geq 0\}$, where each $\mathrm{BA}(n,t)$ is either $\emptyset$ or an infinite set $\{\mathrm{BA}(P,t) : |P| = n, P \subset U\}$, where each $\mathrm{BA}(P,t)$ is a Byzantine agreement protocol for P and t, and $\mathrm{BA}(P,t)$ and $\mathrm{BA}(P',t)$ are copies of each other (i.e., they are the same protocol except for renaming of processors).

We now define complexity measures for Byzantine agreement protocol families (and now it makes sense to think about asymptotic bounds, as functions of t). We do not need analogous definitions for Byzantine broadcast.

The *processor complexity*, $P(t)$, of protocol family BA is the minimum n such that $\mathrm{BA}(n,t) \neq \emptyset$. The *round complexity*, $R(t)$, of protocol family BA is the function describing the round complexity of protocols in the set $\mathrm{BA}(n,t)$, where n is the smallest such that $\mathrm{BA}(n,t) \neq \emptyset$. The *message bit complexity*, $M(t)$, of protocol family BA is the function describing the message bit complexity of protocols in the set $\mathrm{BA}(n,t)$, where n is the smallest such that $\mathrm{BA}(n,t) \neq \emptyset$. The *message size complexity*, $S(t)$, of protocol family BA is the function describing the message size complexity of protocols in the set $\mathrm{BA}(n,t)$, where n is the smallest such that $\mathrm{BA}(n,t) \neq \emptyset$. The *last-round senders complexity*, $M(t)$, of protocol family BA is the function describing the last-round senders complexity of protocols in the set $\mathrm{BA}(n,t)$, where n is the smallest such that $\mathrm{BA}(n,t) \neq \emptyset$.

4. COMPONENT TRANSFORMATIONS

Our general transformation is the composition of two transformations, the first from a Byzantine agreement protocol to a Byzantine broadcast protocol, and the second from a collection of Byzantine broadcast protocols to a Byzantine agreement protocol. In this section we define these component transformations.

4.1 Constructing a Byzantine Broadcast Protocol from a Byzantine Agreement Protocol

In [1] there is a general transformation from Byzantine Agreement to Byzantine Broadcast. This transformation works for any Byzantine agreement protocol, but adds one round. In this subsection we give a more limited, but cheaper, transformation from Byzantine agreement to Byzantine broadcast. The transformation adds no rounds, but it only works for Byzantine agreement protocols that satisfy the following two conditions. (1) In the last round, each correct processor sends precisely the same message (possibly no message) to all processors. (2) The decision of a correct processor must be calculated by applying a fixed function to the multiset of messages (with sender identities removed) received in the last round. All of the protocols to which we will apply our transformation satisfy these conditions.

The transformation (Transformation 1) is as follows. All processors in the committee run the Byzantine agreement protocol and all other processors do nothing. In the last round, each processor in the committee sends its message (possibly no message) to all processors, not just those in the committee. Processors in the committee reach the same decision in the new Byzantine broadcast protocol that they were to have reached in the Byzantine agreement protocol. A processor outside the committee reaches its decision applying the (fixed) decision function of any processor in the committee to the multiset of messages that it itself receives in the last round.

Choose a set of processors P, a subset C of P, and a nonnegative integer t' such that $|C| > 2t'$. Given Byzantine agreement protocol BA for C and t' with round complexity R, message bit complexity M, message size complexity S, and last-round sender complexity L, we construct a Byzantine broadcast protocol for P, C, and t'. This construction is given as Transformation 1. Let BB be the result of applying Transformation 1 to BA.

```
Code for processor p in P:
    Initialization code:
            bb-in(p) is initialized to the input of processor p; bb-out(p) is initialized to $\perp$
    Code for round 1:
            if p is in C then call BA(rounds: R, input: bb-in(p), output: ba-out(p))
                else do nothing
    Code for round r, $2 \leq r \leq R$:
        comment: processors in C continue the subprotocol; the rest continue to do nothing
    Code added to the send phase of round R:
            if p is in C then send to each processor in $P - C$ the same message sent to
                processors in C
    Code added to the receive phase of round R:
            if p is in $P - C$ then receive messages from processors in C
    Code added to the compute phase of round R:
            if p is in $P - C$ then bb-out(p) := the result of applying the decision function
                of a processor in C to the messages just received
        comment: bb-out(p) is p's output
```

Transformation 1: Changing Byzantine Agreement to Byzantine Broadcast

Theorem 1: *BB is a Byzantine broadcast protocol for P, C, and t', with complexities n processors, R rounds, $M + (n - |C|) \cdot L \cdot S$ message bits, S message size, and L last-round senders.*

Proof: By inspection. □

4.2 Constructing a Byzantine Agreement Protocol from Several Byzantine Broadcast Protocols

This subsection describes a simple way to transform a collection of Byzantine broadcast protocols into a Byzantine agreement protocol for P and t. A collection of disjoint subsets of processors is chosen (when the protocol is written, not executed). For each subset in the collection, all the processors execute the following block of rounds. First, two rounds of preliminary message exchanges are performed, during which each processor in P computes a value to use as input to a Byzantine broadcast protocol and, in addition, may compute a value that it currently favors. No two nonfaulty processors favor different values, and if any nonfaulty processor favors a value, then this value is the common input of all nonfaulty processors in the Byzantine broadcast protocol. Then the processors in the subset execute a Byzantine broadcast protocol using the inputs just calculated. The result of the Byzantine broadcast protocol becomes the favored value for any nonfaulty processor that lacked one. The number of subsets is chosen to be large enough to ensure that at least one of the subsets includes a sufficiently large number of nonfaulty processors. After the block of rounds using such a subset, all the processors have the same favored value; this value persists until the end of the protocol, at which time it is chosen as the output for the Byzantine agreement.

Fix the following for the rest of Subsection 4.2:

- an integer t;
- a set of processors P, $|P| = n \geq 3t + 1$;
- an integer B, $1 \leq B \leq t + 1$;
- integers t_b, where $t_b = \lfloor (t + 1 - b)/B \rfloor$, for $1 \leq b \leq B$;
- B disjoint subsets of P, C_b, $1 \leq b \leq B$; and
- Byzantine broadcast protocols BB_b for P, C_b, and t_b, with round complexity R_b, message bit complexity M_b, message size complexity S_b, and last-round senders complexity L_b, $1 \leq b \leq B$.

A few comments are in order. The choice of the C_b's is constrained by the required existence of the BB_b's. The requirement that $|P| \geq 3t + 1$ is necessary for the overhead rounds to work correctly. As long as there are at most t faulty processors in P, the choice of the t_b's ensures that there will be at least one BB_b that operates correctly (i.e., enough processors in C_b are nonfaulty). This follows because $(\Sigma_{b=1}^{B} t_b) + B - 1 = t$, as will be shown below in Lemma 1.

We now construct a Byzantine agreement protocol for P and t, using the BB_b's. Rounds are numbered using ordered pairs (b, i), where $1 \leq b \leq B$ and $1 \leq i \leq R_b + 2$; the right-hand element of the pair increases faster.

```
Code for processor p in P:
    Initialization code:
                favor(p) is initialized to the input to processor p; ba-out(p) is initialized to $\bot$
    Code for rounds $(b,1)$ and $(b,2)$, $1 \leq b \leq B$:
           send: send favor(p) to each processor in P
        receive: receive messages from processors in P
        compute: values(p) := the multiset of values in the messages just received
                 common(p) := the most common value in values(p) (break ties by picking default)
                 number(p) := the cardinality of common(p) in values(p)
                 if number(p) $\geq n - t$ then favor(p) := common(p) else favor(p) := $\bot$
    Code for round $(b,3)$, $1 \leq b \leq B$:
                 call $\mathrm{BB}_b$(rounds: $R_b$, input: common(p), output: bb-out(p))
    Code for rounds $(b,4)$ through $(b, R_b + 2)$, $1 \leq b \leq B$:
        comment: processors continue the subprotocol
    Code added to the end of the compute phase of round $(b, R_b + 2)$:
                 if favor(p) = $\bot$ then favor(p) := bb-out(p)
                 if $b = B$ then ba-out(p) := favor(p)
        comment: ba-out(p) is p's output
```

Transformation 2: Changing Byzantine Broadcasts to Byzantine Agreement

Lemma 1: *For any positive integer B and any $t \geq B - 1$, $\Sigma_{b=1}^{B} \lfloor (t + 1 - b)/B \rfloor = t + 1 - B$.*

Proof: Choose nonnegative integers x and y such that $t = xB + y$ and $0 \leq y < B$.

$\Sigma_{b=1}^{B} \lfloor (t + 1 - b)/B \rfloor = \Sigma_{b=1}^{B} \lfloor (xB - B + B + y + 1 - b)/B \rfloor$

$= xB - B + \Sigma_{b=1}^{B} \lfloor (B + y + 1 - b)/B \rfloor$ since $(xB - B)/B$ is an integer

$= xB - B + \Sigma_{b=1}^{B} \lfloor (b + y)/B \rfloor$

$= xB - B + y + 1 = t + 1 - B$. □

Let BA be the protocol obtained by applying Transformation 2 to BB_b, $1 \leq b \leq B$.

Theorem 2: *BA is a Byzantine agreement protocol for P and t, with complexities*

- *n processors,*

- $\Sigma_{b=1}^{B}(R_b + 2)$ *rounds,*
- $\Sigma_{b=1}^{B}(M_b + 4n^2)$ *message bits,*
- $\max\{S_b : 1 \leq b \leq B\}$ *message size, and*
- L_B *last-round senders.*

Proof: Essentially the same as the proof of a similar transformation (Transformation 2) in [1], except uses Lemma 1 to show that at least on of the BB_b's must have at most t_b faulty processors and thus must satisfy its agreement and validity conditions. Omitted for brevity. □

5. THE NEW BYZANTINE AGREEMENT PROTOCOL

In this section we construct our new Byzantine agreement protocol and calculate its cost. We begin with a transformation from Byzantine agreement back to Byzantine agreement, but with lower costs. Then we apply this transformation recursively.

5.1 Constructing a Byzantine Agreement Protocol From a Byzantine Agreement Protocol

Given positive integer B and Byzantine agreement protocol family BA′ with processor complexity $3t+1$, round complexity $R(t)$, message bit complexity $M(t)$, message size complexity $S(t)$, and last-round senders complexity $L(t)$, we define another Byzantine agreement protocol family BA as follows. (This is Transformation 3.)

We define $BA(P,t)$ for any P and t with $P \geq 3t+1$. Let $t_b = \lfloor (t+1-b)/B \rfloor$, $1 \leq b \leq B$. Choose B disjoint subsets of P, C_b, $1 \leq b \leq B$, such that $|C_b| = 3t_b + 1$. (This is possible because $\Sigma_{b=1}^{B}|C_b| = 3t+1-2B$, by Lemma 1.) Apply Transformation 1 to P and $BA'(C_b, t_b)$ to obtain Byzantine broadcast protocol BB_b for P, C_b, and t_b, for all b, $1 \leq b \leq B$. Apply Transformation 2 to P, t, and BB_b, $1 \leq b \leq B$, to obtain $BA(P,t)$.

Theorem 3: *BA is a Byzantine agreement protocol family with complexities*

- $3t+1$ *processors,*
- $\Sigma_{b=1}^{B}(R(t_b) + 2)$ *rounds,*
- $\Sigma_{b=1}^{B}[M(t_b) + 3(t-t_b) \cdot L(t_b) \cdot S(t_b) + 4(3t+1)^2]$ *message bits,*
- $\max\{S(t_b) : 1 \leq b \leq B\}$ *message size, and*
- $L(t_B)$ *last-round senders.*

Proof: By Theorems 1 and 2. □

5.2 Recursive Construction

Fix an integer $B \geq 2$.

First we define the base protocol family, BA_0. Let LSP be the Byzantine agreement protocol family of [8]. (Actually the problem solved in [8] is for a single processor to distribute its input on which the rest must agree; we consider the obvious transformation of their protocol to the version of Byzantine agreement that we are studying—a copy of the [8] protocol is run for each processor and its input, and processors decide on the majority value in the resulting vector.) Choose any P and t with $|P| \geq 3t+1$. Let $BA_0(P,t)$ be $LSP(P,t)$.

Assume protocol family BA_{i-1} has been defined. We now define protocol family BA_i. Choose any P and t with $|P| \geq 3t+1$. Let $BA_i(P,t)$ be $BA_0(P,t)$ if $t \leq B$, otherwise let $BA_i(P,t)$ be Transformation 3 applied to B and BA_{i-1}. (Lemma 2 below and inspection of the transformations show that BA_{i-1} has the necessary processor complexity for Transformation 3 to be applicable.)

Let BA_i have processor complexity $P_i(t)$, round complexity $R_i(t)$, message bit complexity $M_i(t)$, message size complexity $S_i(t)$, and last-round sender complexity $L_i(t)$, for all $i \geq 0$.

Lemma 2: *BA_0 is a Byzantine agreement protocol family, and*

- $P_0(t) = 3t+1$,
- $R_0(t) = t+1$,
- $M_0(t)$ *is a nondecreasing function bounded above by* $(3t+1)^{t+3}$,

- *$S_0(t)$ is a nondecreasing function bounded above by $(3t-1)^t$, and*
- *$L_0(t) = 3t+1$.*

Proof: By [8]. □

M_0, S_0, and L_0 are only defined for integers. Let $M_*(t) = (3t+1)^{t+3}$, $S_*(t) = (3t-1)^t$, and $L_*(t) = 3t+1$ be functions defined for all real $t \geq 1$. Note that they are all nondecreasing.

Lemma 3: *$R_i(t) = t + 1 + 2 \cdot \Sigma_{j=1}^{i} B^j$ for all i, $i \geq 0$, and all t, $t \geq B^i - 1$.*

Proof: We proceed by induction on i. The basis, $i = 0$, is true by Lemma 2. Suppose the result is true for $i - 1 > 0$. We must show it for i. Pick any $t \geq B^i - 1$.

$R_i(t) = \Sigma_{b=1}^{B}(R_{i-1}(t_b) + 2)$ by Theorem 3

$= \Sigma_{b=1}^{B}(t_b + 1 + 2 \cdot \Sigma_{j=1}^{i-1} B^j) + 2B$ by induction, since each t_b is at least $B^{i-1} - 1$

$= \Sigma_{b=1}^{B} t_b + B + 2B \cdot \Sigma_{j=1}^{i-1} B^j + 2B$

$= t - B + 1 + B + 2B \cdot \Sigma_{j=1}^{i-1} B^j + 2B$, by Lemma 1

$= t + 1 + 2 \cdot \Sigma_{j=1}^{i} B^j$. □

Lemma 4: *$S_i(t) \leq S_*(t/B^i)$ for all i, $i \geq 0$, and all t, $t \geq B^i - 1$. Also, S_i is nondecreasing on $[B^i, \infty)$.*

Proof: We proceed by induction on i. The basis, $i = 0$, is obvious, by Lemma 2.

Suppose the lemma is true for $i - 1 \geq 0$. Choose $t \geq B^i - 1$. By Theorem 3,

$S_i(t) = \max\{S_b : 1 \leq b \leq B\}$

$= S_{i-1}(\lfloor t/B \rfloor)$ since S_{i-1} is nondecreasing by induction

$\leq S_*(\lfloor t/B \rfloor \cdot 1/B^{i-1})$ by induction since $\lfloor t/B \rfloor \geq B^{i-1} - 1$

$\leq S_*(t/B^i)$ since S_* is nondecreasing.

Since $S_i(t) = S_{i-1}(\lfloor t/B \rfloor)$ and S_{i-1} is nondecreasing, it follows that S_i is nondecreasing. □

Lemma 5: *$L_i(t) \leq L_*(t/B^i)$ for all i, $i \geq 0$, and all t, $t \geq B^i - 1$. Also L_i is nondecreasing on $[B^i, \infty)$.*

Proof: Similar to the proof of Lemma 4. □

Let $c_1 = 3$ and $c_2 = 64$.

Lemma 6: *$M_i(t) \leq B^i \cdot M_*(t/B^i) + Bc_1 \cdot t \cdot L_*(t/B^i) \cdot S_*(t/B^i) \cdot i + c_2 \cdot t^2 \cdot \Sigma_{k=1}^{i} 1/B^{k-2}$ for all i, $i \geq 0$, and all t, $t \geq B^i - 1$. Also, M_i is nondecreasing on $[B^i, \infty)$.*

Proof: We proceed by induction on i. The basis, $i = 0$, is obvious.

Suppose the lemma is true for $i - 1$. Pick any $t \geq B^i - 1$. By Theorem 3,

$M_i(t) = \Sigma_{b=1}^{B}[M_{i-1}(t_b) + 3(t - t_b) \cdot t \cdot L_{i-1}(t_b) \cdot S_{i-1}(t_b) + 4(3t+1)^2]$.

By the definition of the t_b's and since M_{i-1}, L_{i-1}, and S_{i-1} are nondecreasing, M_i is also nondecreasing.

We now bound $M_i(t)$. By the definition of the t_b's and since M_{i-1}, L_{i-1}, and S_{i-1} are nondecreasing, we get

$M_i(t) \leq B[M_{i-1}(\lfloor t/B \rfloor) + c_1 \cdot t \cdot L_{i-1}(\lfloor t/B \rfloor) \cdot S_{i-1}(\lfloor t/B \rfloor) + c_2 \cdot t^2]$

$\leq B[M_{i-1}(\lfloor t/B \rfloor) + c_1 \cdot t \cdot L_*(t/B^i) \cdot S_*(t/B^i) + c_2 \cdot t^2]$ by Lemmas 4 and 5

$\leq B[B^{i-1} \cdot M_*(\lfloor t/B \rfloor \cdot 1/B^{i-1}) + Bc_1 \cdot \lfloor t/B \rfloor \cdot L_*(\lfloor t/B \rfloor \cdot 1/B^{i-1}) \cdot S_*(\lfloor t/B \rfloor \cdot 1/B^{i-1}) \cdot (i-1)$

$+c_2 \cdot \lfloor t/B \rfloor^2 \cdot \Sigma_{k=1}^{i-1} 1/B^{k-2} + c_1 \cdot t \cdot L_*(t/B^i) \cdot S_*(t/B^i) + c_2 \cdot t^2]$ by induction

$\leq B[B^{i-1} \cdot M_*(t/B^i) + Bc_1 \cdot (t/B) \cdot L_*(t/B^i) \cdot S_*(t/B^i)(i-1) + c_2 \cdot (t^2/B^2) \cdot \Sigma_{k=1}^{i-1} 1/B^{k-2}$

$+c_1 \cdot t \cdot L_*(t/B^i) \cdot S_*(t/B^i) + c_2 \cdot t^2$ since M_*, L_*, and S_* are nondecreasing

$= B^i \cdot M_*(t/B^i) + Bc_1 \cdot t \cdot L_*(t/B^i) \cdot S_*(t/B^i) \cdot i + c_2 \cdot t^2 \cdot \Sigma_{k=1}^{i} 1/B^{k-2}$. □

5.3 The Protocol

We define the Byzantine agreement protocol family BA as follows. Given P and t, $|P| > 3t$, let $\mathrm{BA}(P,t)$ be $\mathrm{BA}_i(P,t)$, where $i = \lceil \log(4t\log\log t/\log t)\rceil$. All logs are base B.

We need one lemma before the main result.

Lemma 7: For any $t \geq B+1$, $((\log t)/\log\log t)^{(\log t)/\log\log t} < t$.

Proof: Choose $t \geq B+1$.

$((\log t)/\log\log t)^{(\log t)/\log\log t} = B^{((\log t)/\log\log t)\cdot\log((\log t)/\log\log t)}$

$< B^{((\log t)/\log\log t)\cdot\log\log t}$

$= B^{\log t} = t.$ □

Theorem 4: *BA is a Byzantine agreement protocol family with*

- *processor complexity* $P(t) = 3t+1$,
- *round complexity* $R(t) = t + o(t)$,
- *message bit complexity* $M(t) = O(t^2)$, *and*
- *message size complexity* $S(t) = O(t^{1/2})$.

Proof: Correctness follows from the correctness of the constituent transformations.

Let $i = \lceil \log(4t\log\log t/\log t)\rceil$. Note that $t \geq B^i - 1$ (as long as t is large enough so that $\log t \geq 4B\log\log t$).

$P(t)$ is obviously $3t+1$.

$R(t) = R_i(t) = t + 1 + 2\cdot\Sigma_{j=1}^{i} B^j$ by Lemma 3

$\leq t + 1 + 2B^{i+1} = t + 1 + 2B\cdot B\cdot 4t\log\log t/\log t$

$= t + o(t)$.

$M(t) = M_i(t) \leq B^i \cdot M_*(t/B^i) + Bc_1\cdot t\cdot L_*(t/B^i)\cdot S_*(t/B^i)\cdot i + c_2\cdot t^2\cdot \Sigma_{k=1}^{i} 1/B^{k-2}$ by Lemma 6.

We now show that each of the three terms is $O(t^2)$. The third term obviously is.

The first term is $B^i \cdot M_*(t/B^i)$. Note that $B_i \leq B\cdot 4t\log\log t/\log t$. By definition, $M_*(t/B^i) \leq (4t/B^i)^{2t/B^i}$. Substituting for i gives $(\log t/\log\log t)^{(\log t)/(2\log\log t)}$. By Lemma 7, this is $O(t^{1/2})$. Thus the first term is $O(t^{3/2}\log\log t/\log t) = o(t^2)$.

The second term is $Bc_1\cdot t\cdot L_*(t/B^i)\cdot S_*(t/B^i)\cdot i$. Note that $i = o(\log t)$. By definition, $L_*(t/B^i) \leq 4t/B^i = o(\log t)$. By definition, $S_*(t/B^i) \leq (4t/B^i)^{2t/B^i}$. As above, this is $o(t^{1/2})$. Thus the third term is $o(t\cdot(\log t)\cdot t^{1/2}\cdot(\log t)) = o(t^2)$.

Now we bound $S(t)$. By Lemma 4, $S_i(t) \leq S_*(t/B^i)$. As calculated above, this is $o(t^{1/2})$. □

We now show how to achieve a round complexity that is uniformly less than $2t+2$. Let $B = 3$ in the recursive construction and use $\mathrm{BA}_0(P,t)$ for all $t \leq 8$. Then each protocol in BA has the following structure: some number of overhead rounds, followed by an instance of BA_0, followed by some number of overhead rounds,..., followed by an instance of BA_0, where each instance of BA_0 consists of at least three rounds and the overhead rounds come in pairs.

We now show that if there are m instances of BA_0 in $\mathrm{BA}(P,t)$, then there are $3(m-1)$ overhead rounds in $\mathrm{BA}(P,t)$. Since $\mathrm{BA}(P,t) = \mathrm{BA}_i(P,t)$ for some i, it suffices to show the result for all i. We proceed by induction on i. For $i = 0$, there is one instance of BA_0 and no overhead rounds. Suppose the result is true for $i-1$. $\mathrm{BA}_i(P,t)$ consists of three instances of BA_{i-1}, containing m_1, m_2, and m_3 instances of BA_0 respectively. Thus $\mathrm{BA}_i(P,t)$ contains $m = m_1 + m_2 + m_3$ instances of BA_0. By induction and the definition of the transformation, $\mathrm{BA}_i(P,t)$ contains $3(m_1 - 1) + 3(m_2 - 1) + 3(m_3 - 1) + 6 = 3(m-1)$ overhead rounds.

Since each instance of BA_0 has at least three rounds in it, there are more BA_0-rounds than overhead rounds in $\mathrm{BA}(P,t)$. A simple induction on i shows that for all i, in each $\mathrm{BA}_i(P,t)$ the number of BA_0-rounds is $t+1$. Since the total number of rounds is the number of BA_0-rounds plus the number of overhead rounds, the total number of rounds is less than $2(t+1)$.

REFERENCES

[1] B. A. Coan and J. L. Welch, "Modular Construction of Nearly Optimal Byzantine Agreement Protocols," *Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing,* pp. 295–306, Aug. 1989.

[2] P. Berman, J. Garay and K. Perry, "Towards Optimal Distributed Consensus," to appear in IEEE Symposium on Foundations of Computer Science, Oct. 1989.

[3] D. Dolev, "The Byzantine Generals Strike Again," *Journal of Algorithms,* vol. 3, no. 1, pp. 14–30, March 1982.

[4] D. Dolev and R. Reischuk, "Bounds on Information Exchange for Byzantine Agreement," *Journal of the ACM,* vol. 32, no. 1, pp. 191–204, Jan. 1985.

[5] D. Dolev, R. Reischuk, and H. R. Strong, "Early Stopping in Byzantine Agreement," IBM Research Report RJ5406, Dec. 1986.

[6] M. J. Fischer and N. A. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency," *Information Processing Letters,* vol. 14, no. 4, pp. 183–186, June 1982.

[7] M. J. Fischer, N. A. Lynch, and M. Merritt, "Easy Impossibility Proofs for Distributed Consensus Problems," *Distributed Computing,* vol. 1, no. 1, pp. 26–39, Jan. 1986.

[8] L. Lamport, R. E. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems,* vol. 4, no. 3, pp. 382–401, July 1982.

[9] Y. Moses and O. Waarts, "Coordinated Traversal: $(t+1)$-Round Byzantine Agreement in Polynomial Time," *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science,* pp. 246–255, Oct. 1988.

[10] M. Pease, R. E. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults," *Journal of the ACM,* vol. 27, no. 2, pp. 228–234, April 1980.

INEFFECTIVENESS OF SYNCHRONOUS COMMUNICATION

Ravi Condamoor *S. Venkatesan*
University of Texas at Dallas

1. INTRODUCTION

In this paper we give an existential proof showing that a distributed system with synchronous communication and asynchronous processors is equivalent to a system in which both the processors and communication is asynchronous. This result, in conjunction with the result in [We], shows that the presence of synchronous processors or synchronous communication alone in a distributed system is futile if the other is absent.

2. COMPUTATIONAL MODEL

The model we use is similar to the one proposed in [Do]. A *protocol* or a *distributed algorithm* is a collection of *local algorithms* at each of the nodes participating in the protocol. Each local algorithm consists of several *steps.* At every step, the distributed system can perform any one of the following three actions:

(i) *Send(p,m,q)*: The result of this operation is that m is put in the channel connecting p and q.

(ii) *Deliver(p,m,q)*: Message m is delivered to processor q if m is removed out of the channel connecting p and q and put in the input buffer of q.

(iii) *Receive(q)*: Processor q receives messages when it reads a set of messages (maybe empty) from its input buffer.

Each global configuration is represented by

$C_i = \{ps_{1i}, ps_{2i}, \ldots, ps_{ni}\} \bigcup \{l_{1i}, l_{2i}, \ldots, l_{mi}\} \bigcup \{b_{1i}, b_{2i}, \ldots, b_{ni}\}$ where ps_{ji} represents the state of processor j at the i^{th} configuration, l_{ki} represents the contents of channel k at the i^{th} configuration and b_{ji} represents the contents of the input buffer corresponding to processor j at the i^{th} configuration. A run r of protocol P is characterized by a sequence of global configurations C_0, C_1,... where C_0 is the initial global configuration of the distributed system.

Let C_i^s be a configuration of the system with synchronous communication and C_j^a be a configuration in the asynchronous communication system. The processors are asynchronous in both the systems.

Configurations C_i^s and C_j^a are said to be *compatible* if

1. $ps^s_{1i} = ps^a_{1j}, \ldots, ps^s_{ni} = ps^a_{nj}$.
2. $l^s_{1i} \subseteq l^a_{1j}, \ldots, l^s_{mi} \subseteq l^a_{mj}$.
3. $b^a_{1j} \subseteq b^s_{1i}, \ldots, b^a_{nj} \subseteq b^s_{ni}$

   where superscripts s and a represent synchronous and asynchronous system respectively.

Two runs r and r are *equivalent* if for all i the i^{th} configurations are compatible.

Lemma 1: For any run r^s of a protocol P in the synchronous communication system, there exists a run r^a of the same protocol P in the asynchronous communication system such that r^s is equivalent to r^a.

Proof: Asynchronous communication does not have any *a priori* bound on message delivery time. On the contrary, the upper bound on the delivery time is fixed in the case of synchronous communication. This makes it possible for the synchronous communication to be regarded as a special case of asynchronous communication. Since the processors in both the systems are asynchronous, it is now intuitively obvious that for every run of a protocol in the synchronous communication system there exists an equivalent run in the asynchronous communication system. A more formal proof can be found in the full paper.

Lemma 2: For any run r^a in the asynchronous communication system, there exists a run r^s in the synchronous communication system such that r^a is equivalent to r^s.

Proof: Since the processors are asynchronous in both the systems, mapping the *send* and *receive* steps from any run in the asynchronous system to a run in the synchronous system should pose no problem. If in the asynchronous system a message m is deivered at a particular step, the same message may or may not be delivered at the same step in the synchronous system as it has an *a priori* bound on delivery time. If the message is not delivered, by continuing the delivery of message until m is ultimately delivered and by making all the processors wait till that time (possible since the processors are asynchronous), the resulting configuration can be made compatible to the corresponding configuration in the asynchronous system. A more formal proof is by induction and can be found in the full paper.

Theorem: A distributed system with asynchronous processors and asynchronous communication is equivalent to a system with asynchronous processors and synchronous communication.

Proof: The proof follows from Lemma 1 and Lemma 2.

From the equivalence proof given in [We], together with our proof, it follows that a distributed system having synchrony in processors alone is equivalent to one having synchrony in communication alone.

3. SUMMARY

We have presented an existential proof for the equivalence of synchronous communication system to that of asynchronous communication system when both have asynchronous processors. Currently, the plausibility of providing a similar proof for the equivalence of a system having synchronous processors and asynchronous communication to that of one with asynchronous processors and asynchronous communication is being studied.

REFERENCES

[Do] Dolev, D., Dwork, C., & Stockmeyer, L., On the minimal synchronism needed for distributed consensus, *Journal of the Association of Computing Machinery*, Vol. 34, pp. 77-97, 1987.

[We] Welch, J., Simulating synchronous processors, *Information & Computation* 74, pp. 159-171, 1987.

An Interval Logic for Reasoning about Timing Properties of Computer Communication Protocols

Ming-Jye Sheu, Chung-Kuo Chang and Chin-Laung Lei
Department of Computer and Information Science
The Ohio State University, Columbus, OH 43210

Abstract

We study the application of interval logics in specification and verification of timing properties for computer communication protocols. We introduce an interval logic RIL that allows us to describe the relative ordering, as well as the relative timing properties of events. Two protocol verification techniques have been developed based on this logic. The first one follows a verification technique based on deductive reasoning. We show as an example that a Virtual Time CSMA protocol satisfies a network wide first-come first-serve property under bounded propagation delay. The second technique applies RIL to a model-theoretic approach of protocol verification. We show an algorithm that checks if a specification of a protocol expressed in RIL formulas is satisfied in the reachability graph of the protocol. This approach is demonstrated by showing that a real-time version of the alternating bit protocol satisfies its bounded delay property.

1 Introduction

It has been a subject of interests to develop formal verification techniques for computer communication protocols. Exhaustive testing or debugging such a protocol does not suffice to establish its correctness. Previous work on protocol verification has concentrated on proving *safety* or *liveness* properties. Safety and liveness properties concern about the relative order of event occurrences in a network, rather than the relative time that these events occur. Recently, more attentions have been focused on developing communication protocols that satisfy *timing* properties. In general, a timing property requires some property be held within a specific deadline , or some property be consistently held if there is an upper bound of the duration between two events. Examples of the former case can be found in real-time systems that requires message delivery to be within a bounded delay. Examples of the latter case include communication protocols designed for broadcast networks whose implementation depends on a specific upper bound on propagation delay to avoid collisions [4]. In order to prove that a protocol satisfies some given timing properties, it requires a formal mechanism to establish the timing relationship between events occurred in a network. In this paper, we apply an interval logic to specify the timing properties of communication protocols. Based on this logic, we develop several techniques that can be used to verify the correctness of protocols.

Our interval logic is an extension of the conventional temporal logic. While variations of temporal logics have been successfully applied to reasoning about safety and liveness properties of concurrent programs and communication protocols [3, 7], usually they did not consider the specific time that such properties hold. We introduce an extension of the linear time temporal logic, called RIL (Real-time Interval Logic), whose formulas are constructed by applying interval operators over temporal modalities. The association of time intervals with temporal formulas allows us to describe the relative ordering, as well as the relative timing properties of events. The objective of our study is to examine the adequacy of applying such a formal logic over protocol verification. In particular, we want to investigate new verification techniques based on the logic which can be automated easily. We illustrate the use of RIL by applying it over two approaches of protocol verification. In the first approach, we use RIL to specify timing properties of a Virtual Time CSMA protocol. Implementation of such a protocol involves direct manipulation of time and clocks, hence its correctness becomes time dependent. We construct a proof showing that this protocol satisfies a network wide first-come first-serve property if the interval between two message arrivals is greater than a bound on propogation delay of the channel. Our study reveals that by using RIL, we can eliminate the need of global time or clock variables in specifications. In the second approach, we extend this method by combining it with the model-checking technique for verification of finite-state concurrent systems [1, 5, 2]. We show that for a specification expressed in a subclass of RIL formulas, a protocol can be proven to satisfy its specification if and only if its reachability graph with associated timing information meets the specification. We develop an algorithm that checks to see if a reachability graph is a model of the specification. From our study, we found two reasons that make the model-checking approach a great improvement over the previous methods based on reachability analysis. First, the reachability analysis cannot be applied to varify timing properties since these properties may not correspond to any reachable state. Second, a model-checking procedure presents a single algorithm for checking all the properties that can be expressed in the logic, rather than using separate algorithms for checking different properties as required in the case of reachability analysis. Our algorithm runs in time linear in the size of the reachability graph of the protocol and the given RIL formulas. It demonstrates the feasibility of constructing efficient automated verifiers for communication protocols.

2 The Specification Language

In this section, we define our specification language *real-time interval logic* (RIL). In general, an RIL formula is constructed by applying boolean connectives, temporal modalities and interval operators to atomic propositions. In RIL, only one temporal modality, the *periodic-until* operator $\mathcal{U}^k$, is defined. Intuitively, the formula $f\mathcal{U}^k g$ asserts that f must be true every k time units as long as g is false The other conventional temporal operators such as eventually ($\Diamond$), always ($\Box$), until ($\mathcal{U}$), and next-time ($\bigcirc$), etc., can be derived from the modality $\mathcal{U}^k$. The temporal operators are used to describe the ordering and periodic properties of events, and the interval operators are used to reason about relative timing and relative speeds of events.

2.1 Syntax

RIL uses a set of basic symbols consisting of constants, variables, function symbols and predicate symbols. A term of RIL is constructed by applying functions to constants, variables and other functions. An interval is defined by a pair of integer expressions $[m,n]$ where $m \leq n$. Intuitively, $[m,n]$ specifies the time interval from m time units to n time units relative to the current time.

The class of formulas in RIL is defined inductively as follows:

1. If p is a k-place predicate symbol and $x_1, x_2, \dots, x_k$ are terms then $p(x_1, x_2, \dots, x_k)$ is a formula.
2. If f and g are formulas then so are $f \wedge g$, $\neg f$.
3. If f and g are formulas and k is an integer expression then $f\mathcal{U}^k g$ is a formula.
4. If $[m,n]$ is an interval and f is a formula then $[m,n]f$ is a formula.

Note that we classify the primitive predicates into two classes: state predicates and state transition predicates. For example, $x = 1$ and $x > y + z$ are state predicates; $\uparrow p$ and $\downarrow p$ are state transition predicates, where $\uparrow p$ (or $\downarrow p$) asserts that state predicate p changes from false to true (or true to false). A state predicate, like $x = 1$, is an assertion whose truth value will not change unless some action relevant to it happens. On the other hand, a state transition predicate, like $\uparrow (x = 1)$, is only true instantaneously. A formula in RIL which does not contain any temporal or interval operators is called an *atomic formula*.

The other connectives can then be defined as abbreviations in the usual way: $f \vee g \equiv_{\text{def}} \neg(\neg f \wedge \neg g)$, $f \Rightarrow g \equiv_{\text{def}} \neg f \vee g$, $f\mathcal{U}g \equiv_{\text{def}} f\mathcal{U}^1 g$, $\Box^k f \equiv_{\text{def}} f\mathcal{U}^k\text{false}$, $\Box f \equiv_{\text{def}} \Box^1 f$, $\Diamond f \equiv_{\text{def}} \neg\Box\neg f$, and $\bigcirc^k f \equiv_{\text{def}} [k,\infty]f$.

2.2 Semantics

We define the semantics of RIL formulas with respect to a labeled state-transition graph. Formally, a *structure* $M = (S,T)$ for RIL consists of a finite set of states S, and a transition relation $T \subseteq S \times N \times S$ which specifies the possible transitions between states. A *state* $s \in S$ is a mapping which assigns values to variables and predicates. Intuitively, a *transition* $(s,t,s') \in T$ means that state s can reach state s', and it takes t time units. In particular, s_0 is the initial state of M.

A *computation* x of M is a (possibly finite) sequence $< (s_0,t_0),(s_1,t_1),(s_2,s_2),\dots >$ such that $(s_i, t_{i+1} - t_i, s_{i+1}) \in T$, for all $i \geq 0$. We use x^i to denote the suffix of x beginning at (s_i,t_i). We write $M, x \models f$ to mean that the computation x in structure M meets the the specification f. The $\models$ relation is defined inductively as follows:

1. $M, x \models p(x_1, x_2, \dots, x_k)$ iff $s_0(p(x_1, x_2, \dots, x_k))$, i.e. $p(x_1, x_2, \dots, x_k)$ is true at s_0.
2. $M, x \models f \wedge g$ iff $M, x \models f$ and $M, x \models g$,
   $M, x \models \neg f$ iff not $(M, x \models f)$.
3. $M, x \models f\mathcal{U}^k g$ iff $\forall j \geq 0 (M, x \models [jk, \infty]f)$ or $\exists i \geq 0 (M, x^i \models g \wedge \forall j (0 \leq jk < t_i - t_0 \Rightarrow M, x \models [jk, \infty]f))$.
4. For $m \leq n$, $M, x \models [m,n]f$ iff one of the following is true:

   (a) $f = p(x_1, x_2, ..., x_k)$ is a state predicate and $M, x^i \models f$, where $i = max\{j : t_j \leq t_0 + m\}$.
   (b) $f = p(x_1, x_2, ..., x_k)$ is a state transition predicate and $\exists i(t_i = t_0 + m \wedge M, x^i \models f)$.
   (c) $f = f_1 \wedge f_2$ and $M, x \models [m,n]f_1 \wedge [m,n]f_2$.
   (d) $f = \neg f_1$ and $M, x \models \neg[m,n]f_1$.
   (e) $f = f_1 \mathcal{U}^k f_2$ and $(\forall j \geq 0(jk + m \leq n \Rightarrow M, x \models [jk+m, n]f_1)$ or $\exists i \geq 0(m \leq t_i - t_0 \leq n \wedge M, x \models [t_i - t_0, n]f_2 \wedge \forall j(0 \leq jk + m < t_i - t_0 \Rightarrow M, x \models [jk+m, n]f_1)))$.
   (f) $f = [m', n']f_1$ and $M, x \models [m + m', m + n']f_1$.

5. For $m > n$, $M, x \models [m,n]f$, $\forall f$.

Rules 1 and 2 are straightforward. Rule 3 says that $f\mathcal{U}^k g$ is true if and only if f is true every k time units before g becomes true. Rules 4(a) and 4(b) distinguish state predicates from state transition predicates over a time interval. Rule 4(e) gives the interpretation of restricting temporal modalities to time intervals. Rule 4(f) deals with nested intervals. It reflects that inner intervals are interpreted relative to the starting time of their surrounding intervals.

The following theorem shows that since an RIL formula is interpreted relative to the initial time of a given computation, its validity is irrelevant to the initial time of the computation.

Theorem 2.1 (Time Shifting) *If $x =< (s_0, t_0), (s_1, t_1), (s_2, t_2), ... >$ is a computation of a structure $M = (S, T)$, then*

1. *$y =< (s_0, t_0^{'}), (s_1, t_1 + t_0^{'} - t_0), (s_2, t_2 + t_0^{'} - t_0), ... >$ is also a computation of M, and*
2. *For every RIL formula f, $M, x \models f$ iff $M, y \models f$.*

2.3 Axioms

Combining interval operators with logical connectives introduces a set of axioms that can help reduce complex RIL formulas. In the following discussion, we use f, g to represent RIL formulas, and $[m,n]$ to represent an interval operator. RIL satisfies all the tautologies of the first order logic. Also,
(A1) $[m,n]f \wedge [m,n]g \equiv [m,n](f \wedge g)$
(A2) $[m,n]f \rightarrow [m,n]g \equiv [m,n](f \rightarrow g)$
(A3) $[m,n]f \wedge [m,n](f \rightarrow g) \Rightarrow [m,n]g$
(A4) $f \equiv [0,\infty]f$
(A5) $[m,n][m',n']f \equiv [m+m', m+n']f$
(A6) $m \leq l < n \rightarrow ([m,l]\Box f \wedge [l+1,n]\Box f \Leftrightarrow [m,n]\Box f)$
(A7) $m \leq l < n \rightarrow ([m,n]\Diamond f \Leftrightarrow [m,l]\Diamond f \vee [l+1,n]\Diamond f)$

A4 states that any RIL formula can be rewritten in a form with an interval. A5 shows how to combine the nested inverval operators. A6-A7 state that under some circumstances, an interval can be split into two subintervals, or *vice versa*.

3 Protocol Verification via Deductive Reasoning

In this section, we demonstrate as an example by applying RIL to specify the timing properties of a communication protocol and establish a deductive proof to show that the protocol satisfies its specification.

3.1 The Virtual Time CSMA Protocol

The Virtual Time CSMA protocol proposed by Molle utilizes variable rate clocks to control accesses to the shared channel [6]. Unlike other CSMA protocols, it achieves a network wide first-come first-serve scheduling discipline in the ideal case of zero propagation delay. We use RIL to specify the desirable properties of the Virtual Time CSMA protocol and prove that it satisfies the FCFS property.

The protocol is modeled by a program that consists of several station processes running in parallel. Station processes exchange messages over a shared broadcast channel. The channel is modeled by a channel process. Each station process consists of a finite state machine associated with a message queue and two clocks: the real-time clock and the virtual-time clock. A station process also contains a boolean variable *busy* which is shared between the channel and the station process. The boolean variable *busy* is set by the channel process to be true if some message is in transit along the channel. The station process constantly checks this variable to see if the channel is busy. The message queue contains the messages arrived at the station and yet to be sent out to its receiver. Initially, both clocks are reset to zero. The real-time clock runs at a constant rate, while the virtual-time clock runs as follows. When the channel is busy, the station process suspends its virtual-time clock. When the channel is idle, the virtual-time clock is allowed to run. If the virtual-time clock is fallen behind the real-time clock, then the virtual-time clock runs η times faster than the real-time clock until it catches up the real-time clock. Figure 1 shows a finite state machine that implements a station process of the virtual time CSMA protocol. Each transition in the machine is represented as a guarded command $B \rightarrow p$, which means that the machine will take the transition if and only if the condition B is satisfied at the current state and the operation to be executed along the transition is p.

3.2 Specification of the Protocol in RIL

The behaviour of the station processes and the shared channel in the protocol can be specified by a set of axioms expressed as RIL formulas. In the following, let n be the total number of station processes in the network. For all i, $1 \leq i \leq n$, let m_i be a message arrived at station process i. Define $arrivetime(i, m_i)$ to be the value of the real-time clock at process i when m_i arrives. Also, define $transmittime(i, m_i)$ to be the value of the real-time clock when process i sends m_i, and m_i is transmitted successfully to its receiver.

The following predicates describe the current status of the station processes and the channel.

- $send(i, m_i)$ is true iff process i is sending m_i at the current time.
- $\uparrow send(i, m_i)$ is true iff process i starts to send m_i at the current time.
- $\downarrow send(i, m_i)$ is true iff process i completes sending of m_i at the current time.
- $\uparrow busy_i$ is true iff process i senses the channel changes from idle to busy at the current time.
- $\downarrow busy_i$ is true iff process i senses the channel changes from busy to idle at the current time.

The behaviour of the station processes and the channel can now be described in terms of these predicates.

Axioms for the Shared Channel

Let a be the maximum propagation delay of the channel and m_i be a message arrived at station i.

(Ia) $\forall i, collision(i, m_i) \equiv_{def} \uparrow send(i, m_i) \wedge \exists m_j, j \neq i, [-a+1, a-1]\Diamond \uparrow send(j, m_j)$
(Ib) $\uparrow send(i, m_i) \wedge \neg collision(i, m_i) \rightarrow \bigcirc \uparrow busy_i \wedge \forall j \neq i, [1, a]\Diamond \uparrow busy_j$
(Ic) $\downarrow send(i, m_i) \rightarrow \bigcirc \downarrow busy_i \wedge \forall j \neq i, [1, a]\Diamond \downarrow busy_j$
(Id) $\uparrow busy_i \rightarrow \forall j \neq i, \Diamond busy_j, \quad \downarrow busy_i \rightarrow \forall j \neq i, \Diamond \neg busy_j$

Axioms (Ia) defines a collision. I(b) and (Ic) specify the channel delay. (Id) states that once a process senses the channel to become busy (idle), all the processes will eventually sense the channel to be busy (idle).

Axioms for the Station Processes

For all i, let rtc_i be the real-time clock of process i and vtc_i be the virtual-time clock of process i.

(IIa) $\forall t \geq 0, rtc_i = t \rightarrow [1, 1]rtc_i = t + 1$
(IIb) $\forall t, t' \geq 0, rtc_i = t \wedge vtc_i = t' \wedge \neg busy_i \rightarrow [1, 1]vtc_i = min(t' + \eta, t + 1)$
(IIc) $\forall t \geq 0, vtc_i = t \wedge busy_i \rightarrow [1, 1]vtc_i = t$
(IId) $vtc_i - \eta < arrivetime(i, m_i) \leq vtc_i \wedge \neg busy_i \leftrightarrow \uparrow send(i, m_i)$
(IIe) $\forall t \geq 0, rtc_i = t \wedge \uparrow send(i, m_i) \wedge \neg collision(i, m_i) \leftrightarrow transmittime(i, m_i) = t \wedge \Diamond \downarrow send(i, m_i)$

Axioms (IIa) to (IIc) describe the behaviour of the real-time clock and the virtual-time clock. (IIa) states that the real-time clock ticks at every time unit. (IIb) says that when the channel is not busy, the virtual-time clock runs η times faster than the real-time clock. (IIc) says that if the channel is busy, then the virtual-time clock stops. Axioms (IId) and (IIe) state the properties of the messages sent over the channel.

3.3 Verification of the Virtual Time CSMA Protocol

Safety: The safety property of the protocol requires a message arrived at one of the station processes be successfully transmitted to the receiver, i.e., the sending process completes the sending, if no collisions occurred while sending the message. Lemma 3.1 shows the necessary condition for a message arrived at a process to encounter a collision while the message is being sent out by the process.

Lemma 3.1 *For all process i, let m_i be a message arrived at process i.*

$$collision(i, m_i) \rightarrow \exists m_j, j \neq i \wedge \mid arrivetime(i, m_i) - arrivetime(j, m_j) \mid \leq a\eta$$

Theorem 3.2 (Safety) *For any process i, if $\forall j \neq i$,*

$$\begin{aligned} & \mid arrivetime(i, m_i) - arrivetime(j, m_j) \mid > a\eta \\ \rightarrow \quad & \exists t \geq arrivetime(i, m_i) \wedge transmittime(i, m_i) = t \wedge \Diamond \downarrow send(i, m_i) \end{aligned}$$

FCFS: If a message m_i arrives at process i before another message m_j arrives at process j and no collisions occur while process i sends m_i or process j sends m_j, then m_i will be transmitted successfully to its receiver before m_j is transmitted to its receiver. This property can be formally stated as follows.

Theorem 3.3 (FCFS) *For any process i and j where $i \neq j$,*

$$\begin{aligned} & (arrivetime(i, m_i) < (arrivetime(j, m_j) - a\eta)) \wedge \neg collision(i, m_i) \\ \rightarrow \quad & transmittime(i, m_i) < transmittime(j, m_j) \end{aligned}$$

The proof is done by contradiction. From Axiom (IIe) we know that a station process sends a message whenever its virtual-time clock's value is greater than or equal to the timestamp of this message. Suppose message m_j has timestamp value greater than message m_i, and m_j is transmitted before m_i. This implies the virtual-time clock value of j is greater than or equal to the timestamp of m_j, which is greater than the timestamp of m_i. Since m_j is transmitted before m_i, the virtual-time clock value of i when m_j is transmitted is less than the virtual-time clock value of i when m_i is transmitted. This leads to the conclusion that m_j has timestamp value less than m_i's timestamp value. This contradicts the hypothesis. Hence m_j is transmitted after m_i.

4 Protocol Verification via Model Checking

In this section, we apply RIL in a model-theoretic approach to verifying communication protocols. A protocol is represented by a reachability graph that consists of all the possible execution paths of the protocol. We show an algorithm that dertermines whether the reachablility graph of a given protocol is a model of the specification expressed in RIL formulas. For simplicity, we concentrate our discussions on a restricted class of RIL formulas. We define RIL$_c$ to be a subset of RIL such that every formula f in RIL$_c$ has the following two constraints: (1) every interval $[m, n]$ in f is a pair of non-negative integer constants, and (2) if there is a temporal operator $f_1 \mathcal{U}^k f_2$ in f, then k must be equal to 1. Later we show an algorithm which checks if a protocol satisfies its specificatin written in RIL$_c$ formulas.

4.1 The Model

A protocol is modeled by a network of processes connected by point-to-point communication channels. Each process has a local clock and executes a time-driven algorithm that consists of a sequence of operations. We assume that each of these operations has bounded execution time. We also assume that a message, sent by the sending process, will arrive the receiving process within a maximum delay or it will be lost. With these assumptions, we formally define a network of real-time processes as follows. A *real-time process* is a tuple $M = (S, T, s^0)$, where

- S is a finite set of states,
- $T \subseteq S \times E \times N \times S$ is a transition relation such that E is a set of operations $\{-g/C, +g/C, \mathcal{I}\}$ where g is a message, C is a channel connected to M, and $\mathcal{I}$ is an internal operation. Also, N is the set of natural numbers, and
- $s^0 \in S$ is the initial state.

A real-time process can be characterized by its set of reachable configurations as follows. A *local configuration* of M is a tuple (s,t) where $s \in S$ is the current state of M, and $t \in N$ is the current time according to its local clock. Also, (s^0, t_0) is the initial local configuration of M, where t_0 is a given specific time. A local configuration (s', t') is said to *follow* another local configuration (s,t) of M, written as $(s,t) \xrightarrow{M} (s',t')$, iff there exists a transition (s, e, τ, s') in T such that $t' = t + \tau$. A local configuration (s,t) is said to be *reachable* iff there exists local configurations $c_0, \ldots, c_k$ such that $c_0 = (s^0, t_0)$, $c_k = (s,t)$, and c_{i+1} follows c_i for $i = 0, ..., k-1$. Also, $c_0, \ldots, c_k$ is called a *local execution* of M.

A *network of real-time processes* is a tuple $R = (M_1, \ldots, M_m, C_1, \ldots, C_n)$, where $\forall i,\ 1 \le i \le m$, $M_i = (S_i, T_i, s_i^0)$ is a real-time process, and $\forall j,\ 1 \le j \le n$, C_j is a unidirectional channel connecting a pair of processes in R. Also, there are two constants, Δ and d, associated with network R which specify the maximum delay and the maximum channel capacity in R, respectively. For simplicity, let us assume that the maximum channel capacity is one in the following discussion. A network can be characterized by a set of executions as follows. A *global configuration* (or simply a *configuration*) of R is a tuple $\Pi =< (s_1, t_1), \ldots, (s_m, t_m), (c_1, \mu_1), \ldots, (c_n, \mu_n) >$, where $\forall i,\ 1 \le i \le m$, s_i is the current state of M_i, and t_i is the current time of M_i according to its local clock; also, $\forall j,\ 1 \le j \le n$, c_j is the channel content of C_j, and μ_j is the time that the message in C_j is sent according to the sender's local clock. For a given configuration Π, we also define $\mathcal{T}(\Pi) = \max\{t_i : 1 \le i \le m\}$. Also, $\Pi_0 =< (s_1^0, 0), \ldots, (s_m^0, 0), (\Lambda, 0), \ldots, (\Lambda, 0) >$ is the initial configuration of R, where Λ denotes the empty string. A configuration $\Pi' =< (s_1', t_1'), \ldots, (s_m', t_m'), (c_1', \mu_1'), \ldots, (c_n', \mu_n') >$ is said to *follow* another configuration $\Pi =< (s_1, t_1), \ldots, (s_m, t_m), (c_1, \mu_1), \ldots, (c_n, \mu_n) >$ written as $\Pi \rightarrow \Pi'$, provided that

1. $\mathcal{T}(\Pi') > \mathcal{T}(\Pi)$,
2. $\forall i, 1 \le i \le m, t_i' = \mathcal{T}(\Pi')$ implies $(s_i, t_i) \xrightarrow{M_i} (s_i', t_i')$, and
3. $\forall i, 1 \le i \le m, t_i' < \mathcal{T}(\Pi')$ implies (i) $(s_i', t_i') = (s_i, t_i)$, and (ii) there exists a local configuration (s,t) in M_i such that $(s_i, t_i) \xrightarrow{M_i} (s,t)$ and $t > \mathcal{T}(\Pi')$.

Condition 1 says that a configuration follows another due to at least one process completes an operation. Conditions 2 and 3 specify the relationship between individual processes and configurations Π and Π'. Either the current operation of the process completes at Π' (Condition 2), or the current operation of the process will be completed later (Condition 3). A configuration Π is said to be *reachable* iff there exist configurations $r_0, \ldots, r_k$ such that $r_0 = \Pi_0$, $r_k = \Pi$, and r_{i+1} follows r_i for $i = 0, \ldots, k-1$. Moreover, the sequence $r_0, \ldots, r_k$ is called an *execution* of R.

Example: We discuss in this example a real-time version of the alternating bit protocol (ABP) and show how to verify that the protocol satisfies its timing constraints. The protocol is modeled as a network of two real-time processes (Figure 2). Process A sends a datum received from an input stream X to process B. Each datum is combined with a sequence bit. Upon receiving a message from process A, process B checks the sequence bit and return a proper acknowledge to A. We use D_i (or A_i) to denote a data message (or an acknowledgement) associated with sequence bit i, where i is 0 or 1. The channels are modeled as single-element buffers that can delay a message or change a message into an "error", but cannot duplicate or lose messages. We assume that every sending operation (denoted by $-$) takes one time unit, every receiving operation (denoted by $+$) takes two time units, and the maximum channel delay of the network is three time units.

It is straightforward to show that $(1,0) \xrightarrow{A} (2,1) \xrightarrow{A} (3,3) \xrightarrow{A} (4,4) \xrightarrow{A} (1,6)$ is a local execution of process A, and $(5,0) \xrightarrow{B} (6,2) \xrightarrow{B} (7,3) \xrightarrow{B} (8,5) \xrightarrow{B} (5,6)$ is a local execution of process B. Also,

$$\begin{array}{lll}
\overset{1}{<} (1,0),(5,0),(\Lambda,0),(\Lambda,0) > \rightarrow & \overset{2}{<} (2,1),(5,0),(D_0,1),(\Lambda,0) > \rightarrow & \overset{3}{<} (2,1),(5,2),(D_0,1),(\Lambda,0) > \rightarrow \\
\overset{4}{<} (2,3),(5,2),(D_0,1),(\Lambda,0) > \rightarrow & \overset{5}{<} (2,3),(5,4),(D_0,1),(\Lambda,0) > \rightarrow & \overset{6}{<} (2,5),(5,4),(D_0,1),(\Lambda,0) > \rightarrow \\
\overset{7}{<} (2,5),(6,6),(\Lambda,0),(\Lambda,0) > \rightarrow & \overset{8}{<} (2,7),(7,7),(\Lambda,0),(A_0,7) > \rightarrow & \overset{9}{<} (2,9),(7,9),(\Lambda,0),(A_0,7) > \rightarrow \\
\overset{10}{<} (2,11),(7,11),(\Lambda,0),(A_0,7) > \rightarrow & \overset{11}{<} (3,13),(7,13),(\Lambda,0),(\Lambda,0) > \rightarrow & \overset{12}{<} (4,14),(7,13),(D_1,14),(\Lambda,0) >
\end{array}$$

is an execution of the network. Note that the projection of an execution on process A or B is always a local execution of that process, but not *vice versa*.□

4.2 Timed Reachability Graphs

In general, a network contains an infinite number of possible executions. In order to decide whether a network satisfies a given specification, it is desirable to find an appropriate finite representation of the network and then apply existing techniques to verify its properties. In this section we define a finite structure, called the timed reachability graph, to represent a network of real-time processes.

A *snapshot* v of r, where r is a configuration of a network R, is a tuple of current local states and current contents of each buffer in r. Particularly, $v_0 = (s_1^0, \ldots, s_m^0, \Lambda, \ldots, \Lambda)$ is the *initial* snapshot. Note that more than one configurations may map to the same snapshot. A *timed reachability graph of network* R is a tuple $G_R = (V, E)$, where

1. V is the set of vertices labeled by snapshots of R, and for each snapshot v, there is exactly one vertex labeled by v.
2. $E \subseteq V \times N \times V$ is a set of directed edges. An edge $e = (v_i, t, v_j) \in E$ iff there exists one execution $\Gamma = r_0, r_1, \ldots, r_{i-1}, r_i, \ldots, r_j, \ldots$ of the network R, where:(a) v_i is the snapshot of $r_i, \ldots r_{j-1}$ and v_j is the snapshot of r_j, and (b) r_{i-1}'s snapshot is not v_i, and (c) $t = \mathcal{T}(r_j) - \mathcal{T}(r_i)$.
3. For each pair of $v_i, v_j \in V$, there is at most one edge $e \in E$ from v_i to v_j and at most one edge $e' \in E$ from v_j to v_i.

Note that the timed reachability graph defined above is actually a structure defined in Section 2. Therefore, a vertex in V is equivalent to a state in a structure, an edge in E is a transition, and a *computation* x of G_R is a sequence $< (v_0, t_0), (v_1, t_1), (v_2, t_2), \ldots >$ where v_0 is the initial snapshot such that $(v_i, t_{i+1} - t_i, v_{i+1}) \in E$, for all $i \geq 0$.

Example (continued): Figure 3 shows the timed reachability graph of the real-time ABP protocol.

A computation $x =< (v_0, t_0), (v_1, t_1), (v_2, t_2), \ldots >$ of G_R is said to *correspond* to an execution $y = r_0, r_1, \ldots$ of a network R, denoted as $x \sim y$, iff x is a computation starts from the initial snapshot with initial time 0, i.e., $t_0 = 0$ and v_0 is a snapshot of r_0, and the order of the snapshorts in x, must be consistent with the order of the global configurations in y. It is easy to see that for each execution of network R there is a computation in R's timed reachability graph G_R which it corresponds to.

Lemma 4.1 *Let R be a network and G_R be a timed reachability graph of R. Then for each execution y of R, there exists a computation x of G_R such that $y \sim x$.*

A network R is said to be *equivalent* to a structure M *under* Σ,written as $R \cong_\Sigma M$, iff for all RIL formula $f \in \Sigma$, it is true that R satisfies $f \Leftrightarrow M \models f$. We have the following lemmas.

Lemma 4.2 *Let R be a network and G_R be the timed reachability graph of R. Also, let f be any RIL$_c$ formula, y be an execution of R, and x be a computation of G_R. Then $y \sim x \rightarrow y \cong_f x$*

Lemma 4.3 *Let R be a network of real-time processes and G_R be a timed reachability graph of R. Also, let Σ be a given specifications of R written as RIL$_c$ formulas. Then $R \cong_\Sigma G_R$.*

4.3 The Model Checker

From Lemma 4.3, a network of real-time processes meets a specification written as RIL$_c$ formulas if and only if its timed reachability graph satisfies the specification. In this section we present a model-checking algorithm that checks to see if a timed reachability graph satisfies a given RIL$_c$ formula.

Given the current time t and an interval $[m, n]$, the procedure *satisfy* checks if all the computations that are starting from state s and are within $[m, n]$ satisfy the *RIL$_c$* formula f. The procedure *satisfy* does a depth-first search of the timed reachablilty graph based on the timing information. Starting from state s, the procedure either recursively calls the successors of s without checking the subformulas of f in current state if t is less than m, or it returns true immediately, if t is greater than n. If the current time t is inside the interval then the subformulas have to be checked. It is possible that the starting time of an interval falls halfway between two states. In this case, we have to either backtrack to the previous state and evaluate the formula if the formula is an atomic formula, or call a separate procedure *halfway* to take care of the timing information.

The procedure *satisfy* must be able to handle seven cases, depending on whether f is an atomic formula or has one of the following forms: $f_1 \wedge f_2$,$[m, n]f_1$,$f_1 \mathcal{U} f_2$, $\neg(f_1 \mathcal{U} f_2)$, $\neg(f_1 \wedge f_2)$, $\neg[m, n]f_1$.

```
procedure satisfy($s,t,f,m,n$)
begin
        if $t > n$ then return true endif
```

```
    if $t < m$ then
        forall $s' \in succ(s)$ do
            $t' := time\_on\_edge(s,s')$;
            if $t + t' > m$ then return $halfway(s,s',m-t,f,m,n)$
            else return $satisfy(s',t+t',f,m,n)$
            endif
        endforall
    else { $m \leq t \leq n$ }
        case $f$ of
        (1)atomic formula: return $s(f)$
        (2)$f_1 \wedge f_2$: return $satisfy(s,t,f_1,m,n) \wedge satisfy(s,t,f_2,m,n)$
        (3)$[m',n']f_1$: return $satisfy(s,0,f_1,m',n')$
        (4)$f_1 \mathcal{U} f_2$:
            if $satisfy(s,t,f_2,m,n) \vee marked(s,f)$ then return true endif
            if $\neg satisfy(s,t,f_1,m,n)$ then return false endif
            $mark(s,f)$;
            for all $s' \in succ(s)$ do
                $t' := time\_on\_edge(s,s')$;
                if $\neg satisfy(s',t+t',f,m,n)$ then return false endif
            endforall
            return true
        (5)$\neg(f_1 \mathcal{U} f_2)$:
            if $satisfy(s,t,f_2,m,n) \vee marked(s,f)$ then return false endif
            if $\neg satisfy(s,t,f_1,m,n)$ then return true endif
            $mark(s,f)$;
            for all $s' \in succ(s)$ do
                $t' := time\_on\_edge(s,s')$;
                if $\neg satisfy(s',t+t',f,m,n)$ then return false endif
            endforall
            return true
        (6)$\neg(f_1 \wedge f_2)$: return $satisfy(s,t,\neg f_1,m,n) \vee satisfy(s,t,\neg f_2,m,n)$
        (7)$\neg[m',n']f_1$: return $satisfy(s,0,\neg f_1,m',n')$
    end case
end satisfy
```

Example(continued): An interesting characteristic of the real-time ABP protocol is that it guarantees a bounded delay on a complete send-receive-acknowledge operation. We show that the ABP protocol satisfies the following property:

$$send(A, D_0) \Rightarrow [0,13]\Diamond(receive(A, A_0) \vee error)$$

which says that after process A sends a message, it will receive the acknowledgement from process B within 13 time units if there is no corruption to the message. We verify this property by using our model-checking algorithm over the timed reachability graph of ABP. The property can be rewritten into a canonical form using only the basic operators in RIL:

$$(\neg(send(A, D_0) \wedge \neg[0,13]\neg((\neg receive(A, A_0) \wedge \neg error)\mathcal{U} false)))\mathcal{U} false$$

For convenience, we name the above formula f. We also define the subformulas of f as follows.

- $f0 \equiv send(A, D_0)$, which is true iff process A sends D_0 at the current state.
- $f1 \equiv error$, which is true iff the message in either channel is corrupted.
- $f2 \equiv receive(A, A_0)$, which is true iff process A receives A_0 at the current state.
- $f3 \equiv [0,13]\neg(\neg receive(A, A_0) \wedge \neg error)\mathcal{U} false$
- $f4 \equiv \neg send(A, D_0) \vee [0,13]\neg(\neg receive(A, A_0) \wedge \neg error)\mathcal{U} false$

The atomic formulas $f0$, $f1$, and $f2$ can be labeled on the timed reachability graph of ABP as shown in Figure 4. The goal of the verification process is to check if $f4$ is true in every state of the timed reachability graph. From Figure 4, only two states, v_0 and v_8, need further checking. All the other states satisfy $\neg f0$, which implies $f4$. In Figure 5, we show all the executions starting from state v_0 within 13 time units. Every execution in Figure 5 satisfies $f3$, and this leads to the conclusion that $f4$ is true at state v_0. Similarly, we can verify f_4 at v_8. Therefore, the timed reachability graph satisfies f.

5 Concluding Remarks

RIL provides a formal mechanism for specifying and verifying timing properties of computer communication protocols. From our study, we show that RIL is an appropriate tool for expressing timing properties of protocols. Also, the discussions in Section 4 present some preliminary results on finding a decision procedure for the logic. Further researches along two directions will be required in order to materialize the results presented in the paper and construct generally useful automated protocol verifiers based on the logic: (1) We need to find efficient decision procedures for (at least, a significant subclass of) RIL, and (2) We have to consider including various notions of time-related fairness into RIL. These extensions allow us to reason about complex timing properties for a variety of protocols in an algorithmic instead of deductive method.

References

[1] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transations on Programming Languages and Systems*, 8(2):244–263, 1986.

[2] E. A. Emerson and C. L. Lei. Modalities for model checking: Branching time strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

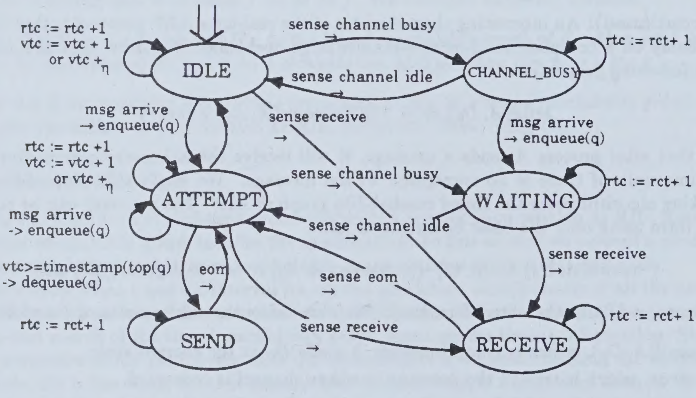
[3] B. T. Hailpern and S. S. Owicki. Modular verification of computer communication protocols. *IEEE Transactions on Communications*, COM-31(1):56–68, Jan. 1983.

[4] P. Jain and S. S. Lam. Modeling and verfication of real-time protocols for broadcast networks. *IEEE Transactions on Software Engineering*, SE-13(8):924–937, Aug. 1987.

[5] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th Annual ACM Symposium on Principles of Programming Languages*, pages 97–107, 1985.

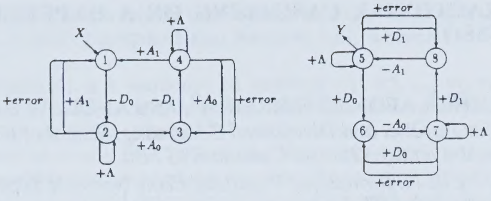
[6] M. T. Molle and L. Kleinrock. Virtual time CSMA: why two clocks are better than one. *IEEE Transactions on Communications*, COM-33(9):919–933, Sept. 1985.

[7] S. S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Transations on Programming Languages and Systems*, 4(3):455–495, 1982.



rtc := rtc +1
vtc := vtc + 1
or vtc +η
sense channel busy →
rtc := rct+ 1
IDLE
sense channel idle ←
CHANNEL_BUSY
msg arrive → enqueue(q)
sense receive →
msg arrive → enqueue(q)
rtc := rtc +1
vtc := vtc + 1
or vtc +η
sense channel busy →
rtc := rct+ 1
ATTEMPT
WAITING
msg arrive -> enqueue(q)
sense channel idle →
sense receive →
vtc>=timestamp(top(q) -> dequeue(q)
eom →
sense eom →
rtc := rct+ 1
rtc := rct+ 1
SEND
sense receive →
RECEIVE

Fig. 1 The Virtual Time CSMA Protocol


X
+Λ
+A₁
1
4
+error
+A₁
−D₀
−D₁
+A₀
+error
2
3
+A₀
+Λ
+error
Y
+D₁
+Λ
5
8
−A₁
+D₀
+D₁
6
7
−A₀
+Λ
+D₀
+error

Process *A* Process *B*

Fig. 2 An Alternating Bit Protocal

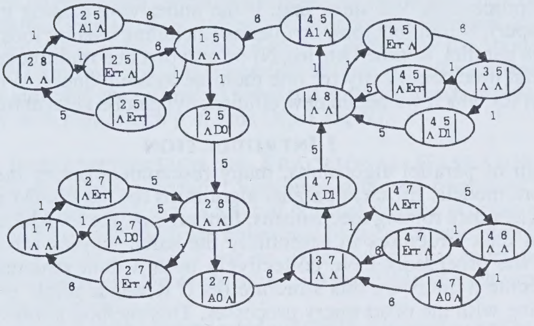

2 5 A1 Λ
2 8 Λ Λ
2 5 Err Λ
2 5 Λ Err
1 5 Λ Λ
2 5 Λ D0
4 5 A1 Λ
4 5 Err Λ
4 5 Λ Err
3 5 Λ Λ
4 8 Λ Λ
4 5 Λ D1
2 7 Λ Err
1 7 Λ Λ
2 7 Λ D0
2 6 Λ Λ
2 7 Err Λ
2 7 A0 Λ
4 7 Λ D1
4 7 Λ Err
4 7 Err Λ
4 6 Λ Λ
3 7 Λ Λ
4 7 A0 Λ
1
5
6
1

Fig. 3 The Timed Reachability Graph of ABP

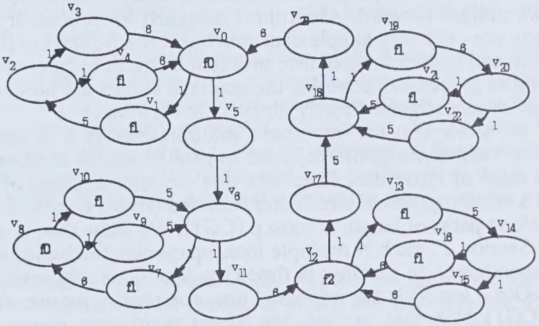

v3
v2
v4
v1
v0
v5
f0
f1
f1
v23
v19
v20
v21
v18
v22
f1
f1
v17
v13
v14
v16
v12
v15
f1
f1
f2
v10
v8
v9
v6
v7
v11
f0
f1
f1

Fig. 4 Timed Reachability Graph Labeld with Atomic Formulas

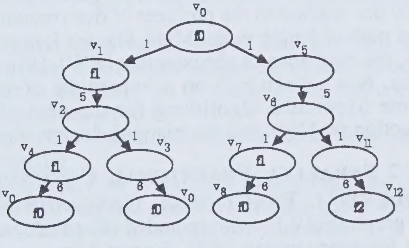

v0
f0
v1
f1
v5
v2
v6
v4
f1
v3
v7
f1
v11
v0
f0
v0
f0
v8
f0
v12
f2
1
5
6

Fig. 5 Executions Starting from v_0 within 13 Time Units

PARALLEL FRACTIONAL CASCADING ON A HYPERCUBE MULTIPROCESSOR

FRANK DEHNE[1], AFONSO FERREIRA[2], AND ANDREW RAU-CHAPLIN[1]
[1] *Center for Parallel and Distributed Computing, School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6.*
[2] *Laboratoire de l'Informatique Parallele, Ecole Normale Superieure de Lyon, 69364 Lyon cedex 07, France.*

ABSTRACT: In this paper we present a parallel implementation of fractional cascading for hypercube multiprocessors. We show that, if the underlying catalog graph is *monotone* (as defined in the paper), N multiple look-up queries (including catalog look-ups) can be executed independently, in parallel, in time $O(t_s \log N)$ on a hypercube multiprocessor of size N, where t_s is the sequential time complexity for one multiple look-up query. We apply our method to various problems for which we obtain new efficient hypercube algorithms.

1 INTRODUCTION

For the design of parallel algorithms, many researchers prefer the PRAM model over processor network models. Efficient parallel algorithms for the PRAM are not constrained by the need for efficient data routing mechanisms; furthermore, the PRAM memory can be used to store and access data structures in essentially the same way as on a standard sequential machine. Once the processors have collectively built a data structure, each of them can individually execute a query on this structure (as if it was a single processor architecture) without interfering with the other query processes. This method allows to apply results from sequential data structures to the design of PRAM algorithms (see, e.g., [ACG], [ACGD], [AG], [DK], [G]).

For processor networks, the parallel execution of independent queries on one joint data structure is not as straight forward. Algorithms designed for processor networks are usually not as elegant; they use only very simple data structures, if any, and are mainly concerned with solving the routing and collision avoidance problem. A more elegant approach is to simulate PRAM algorithms on processor networks; the obtained results are however in most cases less efficient than algorithms designed directly for specific networks.

In this paper we show that for hypercube multiprocessors it is also possible to design elegant yet extremely efficient algorithms based on parallel implementations of data structures.

We study the class of those data structures that are special cases of fractional cascading [CG1]. Consider a catalog graph of size N and bounded (fixed) degree, and a set of N multiple look-up queries along paths of length at most p [CG1]. We show that, if the graph is *monotone* (to be defined in Section 2), such N multiple look-up queries (including catalog look-ups) can be executed independently, in parallel, in time $O(t_s \log N)$ on a hypercube multiprocessor of size N, where $t_s = O(p + \log N)$ is the sequential time complexity for one multiple look-up query as described in [CG1]. Note that, our solution allows an arbitrary number of search queries to access the same node and its catalog at the same time. (This can not be achieved by, e.g., embedding graphs into hypercubes.) The requirement that the graph needs to be monotone is not overly restrictive; such graphs include e.g. all k-nary trees (for fixed k).

In Section 3 we apply our method to the problem of determinating all intersections of n lines with a simple polygonal path of length n; the Multi Slanted Range Search problem is solved in Section 4. In Section 5, we implement a segment tree [BW], [M], [PS] for next element search (n segments, m queries, $N = \max\{n,m\}$) on a hypercube of size N log N. Our approach provides $O(\log^2 N)$ time hypercube algorithms for the next element search problem, the trapezoidal map construction problem, and the triangulation problem.

2 PARALLEL FRACTIONAL CASCADING

2.1 REVIEW OF SEQUENTIAL FRACTIONAL CASCADING

Consider a directed graph G=(V,E) with bounded (fixed) degree; i.e., there exist constants μ_{out} and μ_{in} such that for every vertex $v \in V$, the out-degree and in-degree of v are at most μ_{out} and μ_{in}, respectively. Assume that G is connected and has n vertices (and therefore O(n) edges). As in [CG1], we associate with each vertex v of G a *catalog* C_v consisting of an ordered collection of records from a totally ordered domain U. The graph G together with its catalogs is referred to as a *catalog graph* of size N=max{n,s}, where s is the sum of the sizes of all catalogs.

A *path* π in G (of length p) is a sequence of vertices $v_1, v_2, ..., v_p$ such that for each $1 \leq i < p$, either $(v_i, v_{i+1}) \in E$ [*forward edge*] or $(v_{i+1}, v_i) \in E$ [*backward edge*]. A *multiple look-up query* is a pair (q,π) where q is a value of U and π is a path in G. For each catalog C we denote by $\sigma(q,C)$ the *successor of q in C*, that is the first record of C whose value is greater than or equal to q. The *iterative search problem* consists of executing a multiple look-up query (q,π) by following the path π in G and determining for every vertex v on the path the successor of q in C_v. The path π is assumed to be specified *on-line*, that is the successor v_{i+1} of the vertex v_i in π is only known after the query has reached v_i and determined $\sigma(q,C_{v_i})$.

The above definition of *fractional cascading* follows the one given in [CG1], but assumes fixed degree catalog graphs (instead of catalog graphs with locally bounded degree, see [CG1]) and uses a restricted definition of the considered type of paths. For the above case it follows from [CG1] that in O(N) time and space it is possible to construct, for the standard sequential machine model, a data structure that allows to solve the iterative search problem for one single multiple look-up query with path length p in time O(p + log N).

2.2 A HYPERCUBE IMPLEMENTATION OF FRACTIONAL CASCADING

We now study how to obtain a parallel implementation of fractional cascading on a hypercube multiprocessor of size N; that is a set of N synchronized processors PE(i), $0 \leq i \leq N-1$, where two processors PE(i) and PE(j) are connected by a communication link if the binary representations of i and j differ in exactly one bit. (We assume that every processor has the same constant number of registers; for every register *A* available at each processor, *A(i)* will refer to register A at processor PE(i).)

Compared to the sequential methods presented in [CG1], we need to impose a restriction on the type of catalog graphs. Consider a catalog graph G=(V,E) of size N with n vertices. G is called a *monotone catalog graph* if there exists a (one-to-one) index function *Index*: $V \Rightarrow \{1, ..., n\}$ with the following property: if (v,v') and (w,w') are two edges of G with Index(v)<Index(w) then Index(v')≤Index(w').

In the remainder of this section we will show how, given a *monotone* catalog graph of size N and a set $Q=\{q_1,...,q_N\}$ of N multiple look-up queries along paths of length at most p, these N multiple look-up queries can be executed independantly, in parallel, in time O((p + log N) log N) on a hypercube multiprocessor of size N. We assume, w.l.o.g., that $N=2^d$; all results obtained can be easily generalized. We first give an overview of the algorithm, including the assumed initial configuration of the hypercube, and how the results of the N multiple iterative search queries are reported. We will then present some more details for the different phases of the algorithm.

Algorithm Overview

The graph G is assumed to be stored in the hypercube such that each vertex v with Index(v)=i is stored in register v(i) of processor PE(i). For every vertex v, let *pred(v)* and *succ(v)* be the sets of predesessors and successors in G; i.e., the sets of at most μ_{in} and μ_{out} vertices w such that $(w,v) \in E$ and $(v,w) \in E$, respectively. We assume that register v(i) contains fields v.index(i), v.successor$_1$(i), ..., v.successor$_{\mu_{out}}$(i) and v.predecessor$_1$(i), ..., v.predecessor$_{\mu_{in}}$(i) storing Index(v) and (in sorted order) the indices of the vertices in succ(v) and pred(v), respectively .

We assume that each PE(i) also has a register c(i) to store a catalog record. The catalogs are stored in sorted order with respect to the index of the associated vertices, and each catalog is internally sorted with respect to the order on U. Each record c(i) contains a field c.index(i) storing the index of the associated vertex of G.

Every register v(i) also has a field v.EndCat(i) storing the address (processor number) of the last record of the associated catalog.

The set $Q=\{(q_1,\pi_1), ..., (q_N,\pi_N)\}$ of N multiple look-up queries is given as follows: Every processor PE(i) stores in its register q(i) one arbitrary query value q_j. The search paths π_j ($1 \leq j \leq N$) are determined *on-line* by the following two functions

- start: $U \Rightarrow \{1, ..., n\}$
- g: $V \times U \times U \Rightarrow \{-\mu_{in},...,-1,1,...,\mu_{out}\}$.

For each q_j, start(q_j) is the index of the first vertex v_1 in its path π_j.

Assume that the x^{th} vertex v_x of π_j is stored in register v(i) of processor PE(i), and that the successor $\sigma(q_j, C_{v_x})$ of q_j in C_{v_x} has been determined; let $y = g(v(i), q_j, \sigma(q_j, C_{v_x}))$. If $y<0$ then $v.predecessor_{-y}(i)$ is the index of the $(x+1)^{st}$ vertex v_{x+1} of π_j; otherwise, $v.successor_y(i)$ is the index of the $(x+1)^{st}$ vertex.

It is required that both functions, start and g, can be calculated in constant time.

```
Multi Iterative Search:
(1)  Phase 1: Match every query with the 1st node in its search path and perform the
     respective catalog lookup.
(2)  For x := 2 to p do
(3)       Phase x: Match every query with the xth node in its search path and perform
          the respective catalog lookup.
```

Figure 1: Global Structure of the Multi Iterative Search Algorithm

The global structure of the multi iterative search algorithm is described in Figure 1. The iterative search processes for all N queries $q_1, ..., q_N$ are executed in p phases; each phase moves all queries one step ahead in their search paths.

The general idea is that, in Phase x ($1 \le x \le p$), instead of routing the queries to the respective nodes (possibly resulting in collisions), these nodes are duplicated and routed to the respective queries. In order to obtain the desired time complexity, the algorithm first permutes the queries (in registers q(i)) such that they are sorted with respect to the index of the x^{th} node in their search path. It then creates, in registers v'(i), copies of the respective nodes such that each processor PE(i) containing a query q_j in its register q(i), contains in its register v'(i) a copy of the x^{th} node in the search path of q_j (we will call this a *match* of q_j with the x^{th} node in its search path). Finally, for each node v all queries that have v as the x^{th} vertex in their search path and C_v are merged into one sorted list, and for each query its successor in C_v is determined.

In the following we will present some details of Phase 1 and Phase x ($2 \le x \le p$), respectively. The first phase is different from the remaining phases. When ordering the queries with respect to the index of the first node in their search path, the first phase has to start with an arbitrary permutation of the queries, whereas each subsequent phase will utilize the ordering of the previous phase (in order to improve the time complexity of the algorithm).

The algorithm assumes that, in addition to the registers mentioned above, every processor PE(i) also has a register v'(i) to store another vertex of G as well as other auxiliary registers R(i), N(i), q'(i), q''(i), c'(i), N'(i), N''(i), LS(i), Shift(i) and Dest(i).

Phase 1 of the Multi Iterative Search Algorithm

An outline of Phase 1 is given in Figure 2. The algorithm consists of five steps; consult Appendix A for a describtion of the standard hypercube operations referred to in the figure.

```
Phase 1:
(1)  Every PE(i): N(i):=f(Start,q(i),∞)
(2)  Sort([q(i), N(i)], N(i))
(3)  MoveVerticesToQueries
(4)  SelectCatalogs
(5)  SearchCatalogsForQueries
```

Figure 2. Outline of Phase 1.

First, every processor PE(i) calculates the index of the first node in the search path of its query q(i), and stores this value in the register N(i). Then in Step 2, the queries are sorted by the index of the first node in the search path, i.e. N(i). In Step 3 (see Figure 3), the source nodes are copied to the queries for which they are the first node in their search path. Finally, in Steps 4 and 5 (see Figures 4 and 5, respectively), the catalogs associated with the current vertices are selected and, for each query the successor record in the respective catalog is determined, respectively. Steps 3-5 can be performed in O(log N) time.

```
MoveVerticesToQueries:
(1)  Every PE(i): N'(i):=N(i); Dest(i) :=-1
(2)  Route([N'(i)], i-1, i>0)
(3)  PE(N): N'(N):=N(N) + 1
(4)  Every PE(i) with N'(i)≠N(i): Dest(i):=i
(5)  Route([Dest(i)], N(i), N'(i)≠N(i))
(6)  Every PE(i): v'(i):=v(i)
(7)  RouteAndCopy([v'(i)], Dest(i), Dest(i)≠-1)
```

Figure 3. Sketch of Procedure MoveVerticesToQueries.

```
SelectCatalogs:
(1)  Every PE(i): N'(i):=N(i); Dest(i) :=-1
(2)  Route([N'(i)], i-1, i>0)
(3)  PE(N): N'(N):=N(N) + 1
(4)  Every PE(i) with N'(i)≠N(i): Dest(i):=v'.EndCat(i)
(5)  RouteAndCopy([Dest(i)], Dest(i), Dest(i)≠-1)
```

Figure 4. Sketch of Procedure SelectCatalogs.

```
SearchCatalogs:
(1)  Every PE(i) with Dest(i)≠-1 : c'(i):=c(i)
(2)  Number(H(i), Dest(i)≠-1)
(3)  Concentrate([c'(i),Dest(i)], Dest(i)≠-1)
(4)  Reverse([c'(i), Dest(i)], N, H(i)+N-1)
(5)  BitonicMerge2([q'(i),c'(i),r'(i)],v'index(i),q'(i),c'index(i),c'(i),0,N-1,0,H(i)-1)
(6)  Every PE(i)  : flag'(i):=flag(i)
(7)  Route([flag'(i)], i+1, i<N)
(8)  RouteAndCopy([c'(i)],(i),flag'(i)≠flag(i) and flag(i)=c)
```

Figure 5. Sketch of Procedure SearchCatalogsForQueries.

Phase x (2≤x≤P) of the Multi Iterative Search Algorithm

As indicated above, the purpose of each subsequent phase is to advance, in time O(log N), all queries by one step in their search paths. After Phase x-1 has been completed, all queries are sorted (in registers q(i)) with respect to the index of the (x-1)th node in their search path. Each processor PE(i) contains in its register v'(i) a copy of the (x-1)th node in the search path of the query stored in q(i). In register c'(i), PE(i) stores a copy of the successor catalog element of query q(i) in catalog $C_{v'(i)}$. The desired effect of Phase x is to have all queries sorted (in registers q(i)) with respect to the index of the xth node in their search path, and have each processor PE(i) contain (in its register v'(i)) a copy of the xth node in the search path of query q(i) and (in register c'(i)) a copy of the successor of q(i) in $C_{v'(i)}$.

An outline of the algorithm for Phase x is given in Figure 6. First (in Step 1), every PE(i) computes for the query currently stored in its register q(i) which edge to use for the next step in the search path as well as the index of the next node, and stores these two numbers in the auxiliary registers R(i) and N(i), respectively. Note that if the query has to be routed backward in the graph G then a negative value is stored in the register R(i). In Step 2, all queries are sorted by the index of the next node in their search paths. By sorting first the backward moving queries and then the forward moving queries, this sorting operation can use the properties of the previous permutation of the queries and be performed by a procedure OrderQueriesByNextVertex in time O(log N). Once this ordering has been obtained, the nodes can be matched with the queries, and the respective catalogs can be selected and searched, in time O(log N) in the same way as above.

```
Phase x , $2 \le x \le p$:
(1)  Every PE(i):   R(i):=g(v'(i),q(i),c'(i));
                    If R(i)>0
                        THEN N(i):=v'.successor$_{R(i)}$(i)
                        ELSE N(i):=v'.predecessor$_{-R(i)}$(i)
(2)  OrderQueriesByNextVertex
(3)  MoveVerticesToQueries
(4)  SelectCatalogs
(5)  SearchCatalogsForQueries
```

Figure 6. Overview of Phase x, $2 \le x \le p$.

What remains to be discussed is procedure OrderQueriesByNextVertex. This procedure, which is sketched in Figure 7, creates in time O(log N) the new ordering of the queries with respect to the indices of the next nodes in the search paths.

We first consider all forward edges to next vertices in the search paths; the backward edges are handled analogously. Let (v,w) and (v',w') be two such edges for two queries q and q', respectively, with the property that $g(v,q,\sigma(q,C_v))=g(v,q,\sigma(q,C_{v'}))$. From the monotonicity of G it follows that if $Index(v)<Index(v')$ then $Index(w) \le Index(w')$. Therefore, the subsequence of queries q for which $g(v,q,\sigma(q,C_v))$ has the same value r is already sorted with respect to the index of the next vertex. Furthermore, since each node has an outdegree of at most μ_{out}, there are at most $\mu_{out}=O(1)$ such subsequences. The new ordering of the queries can therefore be created in time $O(\mu_{out} \log N)=O(\log N)$ by extracting these μ_{out} ordered subsequences and mergeing them in μ_{out} bitonic merge steps. For the backward edges the same idea applies because a monotone graph has the same monotonic properties forwards or backwards.

```
OrderQueriesByNextVertex:
(1)  Initialize all shift registers.
(2)  Every PE(i): Shift(i):=0
(3)  FOR r:= $-\mu_{in},\ldots,-1,1,\ldots,\mu_{out}$ DO
(4)      Every PE(i) with R(i)=r: q'(i):=q(i), N'(i):=N(i)
(5)      Concentrate([q'(i), N'(i)], N'(i)=r)
(6)      Number(LS(i), N(i)=r)
(7)      ls := LS(0)
(8)      shift := Shift(0)
(9)      Reverse([q'(i), N'(i)],0,ls)
(10)     Route([q'(i), N'(i)], i+Shift(i), N'(i)=r)
(11)     Every PE(i) with N'(i)=r: q"(i):=q'(i), N"(i):=N'(i)
(12)     BitonicMerge([q"(i), N"(i)],N"(i),0,shift,shift+ls)
(13)     Every PE(i): Shift(i):=Shift(i)+LS(i)
(14) Every PE(i): q(i):=q"(i), N(i):=N"(i)
```

Figure 7. Sketch of Procedure OrderQueriesByNextVertex.

Theorem 1. *For a monotone catalog graph of size N (and bounded degree), N iterative search queries along paths of length at most p can be executed independantly, in parallel, in time O((p + log N) log N) on a hypercube multiprocessor of size N.*

For the algorithm presented above, it is in fact not necessary that all catalogs are given a priori. It is easy to see that for each phase only those catalogs to be accessed by at least one query need to be available, provided that these catalogs are stored in sorted order with respect to the indices of the respective nodes.

3 DETERMINATION OF ALL INTERSECTIONS OF n LINES WITH A SIMPLE POLYGONAL PATH OF LENGTH n

Consider the problem of determining, for a given simple polygonal path P of length n, all intersections with a given query line l. Chazelle and Guibas [CG2] have presented an O(n) space, O(n log n) preprocessing time, and $O(k \log \frac{n}{k})$ query time sequential solution (where k is the number of reported results) based on sequential fractional cascading. They introduce a convex hull decomposition tree T with a root node representing the convex hull of P; recursively, P is then split into two halves represented by each subtree. The convex hull at each vertex is stored as follows: at every leaf, the respective line segment is stored; for every internal node the at most two merge lines for creating the convex hull for this node (from the convex hulls of the two direct children) are stored. Line queries are answered in a branch-and-bound fashion; the subtree H_l of T traversed by a query line l is shown to have at most $O(k \log \frac{n}{k})$ vertices. The decision to be made at each node of T consists mainly of locating the slope of l in a sorted list of slopes of convex hull edges, and hence fractional cascading can be applied.

We observe that for a line l, instead of traversing H_l in a branch-and-bound fashion as in [CG2], we can also traverse H_l in an inorder traversal with exactly the same type of decision to be made at every node. Such a traversal (as well as the tree T) meet the requirements for parallel fractional cascading on a hypercube described in Section 2.

It is easy to see that on a hypercube of size n, the convex hull decomposition tree T can be constructed in time $O(\log^2 n)$. Applying Theorem 1, we obtain

Theorem 2. *Given a simple polygonal path P of length n and a set of n arbitrary query lines, then for all query lines all intersections with P (with a maximum of k results per query) can be determined on a hypercube multiprocessor of size N in time* $O((k \log \frac{n}{k} + \log n) \log n)$.

Note: We assume that at the end of each phase of parallel fractional cascading, every processor can report a result without having to store it. Otherwise, after each phase the reported results need to be concentrated in order to obtain an even data distribution, and the number of processors and time complexity increase to $O(n+M)$ and $O((k \log \frac{n}{k} + \log n) \log (n+M))$, respectively, where M denotes the total output size for all queries. A similar argument holds for Section 4.

4 MULTI SLANTED RANGE SEARCH

Consider a set S of n points in the Euclidean plane and n slanted range queries as described in [CG2]. Chazelle and Guibas present a sequential fractional cascading algorithm that allows to answer one slanted range queries in time $O(k + \log n)$, where k is the size of the output, with $O(n)$ and $O(n \log n)$ space and preprocessing, respectively. Their algorithm is based on a tree T representing S as follows: the root of T represents the lower hull of S; the left and right subtree represent the lower hulls of the left and right half, respectively, of the set S minus its lower hull points. A slanted range search query is again executed by a branch-and-bound type search on T, where the decision at each node reduces to a catalog look-up of the angle of a border segment of the range in the sorted list of angles of the lower hull edges for the that node.

We observe that if we replace this type of traversal by an inorder traversal of length $O(k \log \frac{n}{k})$ we meet the requirement of parallel fractional cascading on a hypercube multiprocessor described in Section 2. Since the convex hull of n points can be obviously computed on a hypercube of size n in time $O(\log^2 n)$ by using the standard divide and conquer approach together with bitonic merging, it follows that T can be computed in time $O(\log^3 n)$. Applying Theorem 1, we obtain

Theorem 3. *Given a set of n points in the Euclidean plane then, with a preprocessing of* $O(\log^3 n)$, *n slanted range search queries [CG2] (with a maximum of k results per query) can be solved on a hypercube of size N in time* $O((k + \log N) \log N)$.

5 A HYPERCUBE IMPLEMENTATION OF A SEGMENT TREE FOR NEXT ELEMENT SEARCH, AND APPLICATIONS

We will now use the results obtained in Section 2 to present an efficient parallel implementation, for the hypercube multiprocessor, of a well known data structure: the segment tree [BW]. The segment tree is a widely used structure which has for example been used to obtain efficient implementations of plane sweep algorithms in computational geometry [BW], [M], [PS]. Here, we consider an application of the segment tree to the next element search problem.

The *next element search* problem is a well known problem in computational geometry. Given a set S of n non intersecting line segments $I_1,\ldots,I_n$ and a direction D_{next} (without loss of generality we will assume that D_{next} is the direction of the positive Y-axis), the next element search problem consists of finding for each point p_i of a set of m query points $p_1,\ldots,p_m$ the line segment I_j first intersected by the ray starting at p_i in direction D_{next} ($m=O(n)$).

A well known method for solving the next element search problem is to apply a plane sweep in direction D_{next} using a *segment tree* [BW], [M], [PS]. Let $I_i^{(x)}$ [$p_i^{(x)}$] be the projection of line segment I_i [point p_i, respectively] onto the x-axis, and let $(x_1,x_2,\ldots,x_{2n})$ be the sorted sequence of the projections of the 2n endpoints of $I_1,\ldots,I_n$ onto the x-axis. The segment tree $T(S) = (V_S,E_S)$ for S is the complete binary tree with leaves $x_1,\ldots,x_{2n}$. For every node v of T(S), an interval xrange(v) is defined as follows:

- if v is a leaf x_i, then xrange(v) = $[x_i,x_{i+1})$. ($[x_{2n},x_{2n+1})=[x_{2n},x_{2n}]$)
- if v is an internal node, then xrange(v) is the union of all intervals xrange(v') such that v' is a leaf of the subtree of T(S) rooted at v.

With every node v of a segment tree T(S) there is associated a node list $NL(v) \subseteq S$ which is defined as follows: $NL(v) = \{ I \in S \mid \text{xrange}(v) \subseteq I^{(x)}$ and not (xrange(father of v) $\subseteq I^{(x)})\}$. For the remainder let h denote the height of T(S).

A segment tree T(S) is a monotone catalog graph with respect to the following index function: assume all nodes being sorted by height as major key and xrange as minor key, then the index of each node is its rank with respect to this ordering.

For every query point p, we define path(p) to be the path in T(S) from the root to the leaf v such that $p^{(x)} \in$ xrange(v). In order to solve the next element search problem, each query point p is routed along path(p). At every node v on the path, the successor of p in NL(v) is determined (this process will be referred to as *locating* p in NL(v)). Hence, the next element search problem for n query points reduces to n multiple look-up queries on T(S). We show next how to build T(S), in particular how to build the node lists (catalogs) NL(v).

Note that each line segment can occur in O(log n) node lists and, thus, the sum of the lengths of all node lists is O(n log n) [M]. Hence, storing the segment tree with all its node lists in a hypercube multiprocessor requires O(n log n) processors.

For a segment $I \in S$ with $I^{(x)} = [a,b]$ we define *l-path(I)* to be the path from the root of T(S) to the leaf v of T(S) with $a \in$ xrange(v). Likewise we define *r-path(I)* to be the path from the root of T(S) to the leaf v of T(S) with $b \in$ xrange(v). We observe that, if a line segement I is contained in a node list NL(v), then exactly one of the following four cases applies:

(1) $v \in$ l-path(I)
(2) v is the right child of a node $v' \in$ l-path(I)
(3) $v \in$ r-path(I)
(4) v is the left child of a node $v' \in$ r-path(I)

We define $NL_r(v)$, $r \in \{1,2,3,4\}$, to be the set of all $I \in NL(v)$ for which case r applies.

The algorithm for constructing the segment tree T(S) consists of four parts. In Part r, $1 \le r \le 4$, all line segment are routed through T(S). When they arrive at the nodes of height i, $1 \le i \le h$, the node lists $NL_r(v)$ of all those nodes are created. In order to efficiently determine, for a query point, the next line segment in a node list NL(v), the segments have to sorted with respect to the above-below relation within the vertical slab defined by xrange(v). We will create every sublist $NL_i(v)$ in sorted order; at the end of Part 4, the node lists NL(v) in sorted order are obtained from the sublists $NL_i(v)$ by applying bitonic merge.

We will show how to execute Parts 1 and 2; Parts 3 and 4 follow by symmetry. We assume a hypercube of size N=max{n,m} where initially every processor stores one line segment and one query point; w.l.o.g., $m=n=N=2^d$.

We first present Part 1 of the algorithm, i.e. how to create the node list $NL_1(v)$ for all nodes v with Level(v)=i.

This problem is solved by taking as the set of queries (for our parallel fractional cascading algorithm) the set of line segments, and route every segment $I \in S$ along l-path(I). At the end of Phase i ($1 \le i \le h$) for every node v with Level(v)=i there exists a block of consecutively numbered processors containing all line segments s such that $v \in$ l-path(s). From these, we can immediately extract all line segments $s \in NL_1(v)$.

What remains to be discusses is how to obtain a sorted ordering of the node lists $NL_1(v)$. We observe that our fractional cascading algorithm applied to a segment tree T(S) is stable in the following sense: if two queries (line segments) q_1 and q_2 are initially stored in processors $PE(j_1)$ and $PE(j_2)$ with $j_1<j_2$, and the i_{th} node in l-path(q_1) is the same as the i^{th} node in l-path(q_2), then at the end of Phase i the queries q_1 and q_2 are stored in two processors $PE(j_1')$ and $PE(j_2')$ with $j_1' < j_2'$. Therefore, we initially sort all line segments by the y-coordinates of their left endpoints. Then, at the end of each Phase i all line segments which were routed to a node v are ordered by y-coordinate; i.e., $NL_1(v)$ is in the desired order. Applying the results from Section 2.2 we therefore obtain the following.

Lemma 1. *Part 1 of the segment tree construction algorithm can be executed in time $O(\log^2 N)$ on a hypercube of size N log N.*

We now turn to Part 2 of the algorithm; i.e., constructing the node lists $NL_2(v)$. As in Part 1, each line segement I in a node list $NL_2(v)$ still has the property that $xrange(v) \subseteq I^{(x)}$, but in contrast to the former case a total ordering of the line segements in $NL_2(v)$ can not be obtained by using the sorted order of the left (or right) endpoints of the segments.

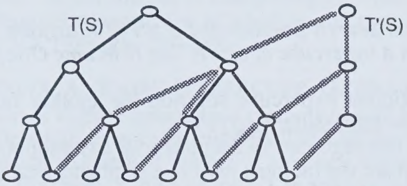


Figure 8. The Trees T(S) and T'(S).

Let T'(S) be the "shifted" version of T(S) as shown in Figure 8. For each line segement $I \in S$ let l-path'(I) be the path in T'(S) either from the last node v of l-path(I), if v is a right child in T(S), or otherwise from the right sibling of v in T(S), to the root of T'(S).

From the above definitions it follows that if a segment $I \in S$ is in a node list $NL_2(v)$ then v is a node in l-path'(I).

We also observe that if l-path'(I)=$(w_1, ..., w_h)$ and $I \notin NL_2(w_i)$, then $I \notin NL_2(w_j)$ for all $j \geq i$. Thus, for three nodes w_0, w_1, and w_2 in T'(S) such that $(w_1,w_0) \in E_S'$ and $(w_2,w_0) \in E_S'$ it follows that $NL_2(w_0) \subseteq NL_2(w_1) \cup NL_2(w_2)$. Let $xrange(w_0)=[a,b)$ and consider the ordering of $NL_2(w_0)$ obtained by sorting the line segments by the y-coordinate of their intersection with the line x=a. This ordering can be constructed from the analogous orderings of $NL_2(w_1)$ and $NL_2(w_2)$ by eliminating from these sequences the elements not contained in $NL_2(w_0)$ and merging the so obtained subsequences.

The idea for Part 2 of the algorithm is to route all line segements along l-path'(I). Since T'(S) is a monotone (catalog) graph with O(n) sources, this can be implemented in time $O(\log^2 N)$. It is easy to see that during this search, it is possible to delete some line segments (i.e., eliminate them from further consideration) in any phase of the parallel fractional cascading algorithm without changing the time complexity. In this particular case, we delete a line segment $I \in S$ if it has been routed to some node v with $I \notin NL_2(v)$.

At the end of Phase i, $1 \leq i \leq h$, for each node w_0 in G with $Level(w_0)=h-i+1$ there exists a consecutive sequence of processors containing all line segments $I \in NL_2(w_0)$. In Phase i-1, these line segments have been routed to at most two different nodes w_1 and w_2. If $NL_2(w_1)$ and $NL_2(w_2)$ where previously ordered as described above, then the same ordering for $NL_2(w_0)$ can be obtained by extracting the two subsequences of segments previously routed to $NL_2(w_1)$ and $NL_2(w_2)$, respectively, and merging these subsequences using a bitonic merge. Since only two line segements where initially routed to every source of T'(S), the orderings of all lists $NL_2(v)$ can be maintained through all phases with an overhead of O(log N) steps per phase.

This yields

Lemma 2. *Part 2 of the segment tree construction algorithm can be executed in time $O(\log^2 N)$ on a hypercube of size N log N.*

Summarizing, we obtain

Theorem 4. *The segment tree construction problem can be solved on a hypercube of size N log N in time $O(\log^2 N)$; $N=\max\{m,n\}$.*

Our strategy for solving the next element search problem has five phases. In phase one, we construct the segment tree T(S) as described above. In phase two, we route the query points down T(S), constructing a query list QL(v) for each node v by storing copies of all queries that visit it. The query lists are stored in the same manner as the catalogs (see Section 2.2). Phase three converts in $O(\log^2 N)$ time the catalogs NL(v) into a forest of binary trees, one for each catalog. Each NL(v), which is sorted as described above, is represented by a balanced binary search tree. This step can be completed by having each catalog element calculate its level and position in its tree. A single $O(\log^2 N)$ sort can then be used to construct the forest of binary trees. Each Catalog lookup can now, in part four, be performed by executing a multiple binary search in which each query in each query list is routed through the respective binary search tree. Finally, in phase five, the partial results generated in phase four, are sorted by the original query points, and the final results are calculated by choosing the best answer for each query point.

Summarizing, we obtain

Theorem 5. *The next element search problem for a set of n disjoint line segments and m query points can be solved on a hypercube of size N log N in time* $O(\log^2 N)$*;* $N=\max\{m,n\}$.

Theorem 5 implies an efficient hypercube solution for another fundamental geometric problem: the construction of the *trapezoidal map* [TW].

Given a set S of n disjoint line segments in the plane; for any endpoint p of a segment in S, the trapezoidal segments for p are the (at most two) line segments first intersected by the rays emanating from p in direction of the positive and negative y-axis, respectively. The construction of the trapezoidal map consists of finding for each endpoint of the segments in S its trapezoidal segments.

This problem is fundamental in computational geometry and is frequently used to solve other geometric problems; see e.g. [G], [TW], and [Y]. Atallah, Cole, and Goodrich [ACG], [G] presented an O(log n) time algorithm for computing the trapezoidal decomposition on a PRAM with O(n) processors (and O(n log n) space). As a consequence of Theorem 5, we obtain

Corollary 1. *For a set of n disjoint line segments, the trapezoidal map can be computed on a hypercube with n log n processors in time* $O(\log^2 n)$.

Yap [Y] has shown that on a PRAM with O(n) processors (and O(n log n) space), the *triangulation* of a simple polygon (see [TW]) can be computed in time O(log n) by essentially applying two calls of the trapezoidal map algorithm (of [ACG], [G]). By combining the result in [Y] with Corollary 1, we obtain

Corollary 2. *An n-vertex simple polygon can be triangulated on a hypercube multiprocessor of size n log n in time* $O(\log^2 n)$.

REFERENCES

[ACG] M.J. Atallah, R. Cole, and M.T. Goodrich, "Cascading divide-and-conquer: a technique for designing parallel algorithms", SIAM J. Comput. 18:3, pp. 499-532, 1989.

[ACGD] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing, and C. Yap, "Parallel computational geometry", Algorithmica 3:3, 1988, pp. 293-327.

[AG] M.J. Atallah, M.T. Goodrich, "Efficient plane sweeping in parallel", in Proc. ACM Symp. Computational Geometry, 1986, pp. 216-225.

[B] K.E. Batcher, "Sorting networks and their applications", in Proc. AFIPS Spring Joint Computer Conference, 1968, pp. 307-314.

[BW] J.L. Bentley and D. Wood, "An optimal worst case algorithm for reporting intersections of rectangles", IEEE Transactions on Computers 29:7, 1980, pp. 571-576.

[CG1] B. Chazelle, L.J. Guibas, "Fractional cascading: I. A data structuring technique", Algorithmica 1:2, 1986, pp. 133-162.

[CG2] B. Chazelle, L.J. Guibas, "Fractional cascading: II. Applications", Algorithmica 1:2, 1986, pp. 163-192.

[DK] N. Dadoun and D.G. Kirkpatrick, "Parallel processing for efficient subdivision search", in Proc. ACM Symp. on Computational Geometry, 1987, pp. 205-214.

[DR] F. Dehne, A. Rau-Chaplin, "Implementing data structures on a hypercube multiprocessor, and applications in parallel computational geometry", Tech. Rep. SCS-TR-152, School of Computer Science, Carleton Univ., Ottawa, Canada K1S5B6.

[G] M.T. Goodrich, "Efficient parallel techniques for computational geometry", Ph.D. thesis, Department of Computer Science, Purdue University, 1987.

[M] K. Mehlhorn, "Data structures and algorithms 3: multi-dimensional searching and computational geometry", Springer Verlag, 1984.

[NS] D. Nassimi, S. Sahni, "Data broadcasting in SIMD computers", IEEE Trans. on Computers 30:2, 1981, pp. 101-106.

[PS] F.P. Preparata and M.I. Shamos, "Computational geometry - an introduction", Springer Verlag, 1985.

[TW] R.E. Tarjan and C.J. Van Wyk, "An O(n log log n) time algorithm for triangulating a simple polygon", SIAM Journal of Computing 17, 1988, 143-178.

[Y] C.-K. Yap, "Parallel triangulation of a polygon in two calls to the trapezoidal map", Algorithmica 3:2, 1988, pp. 279-288.

APPENDIX A: BASIC HYPERCUBE OPERATIONS

The following is a list of slightly generalized versions of well known hypercube operations, as used in Section 2. (In addition to those registers listed below, the actual implementation of these operations may require a constant number of auxiliary registers.)

Rank(Reg(i),Cond(i)): Compute, in time O(log N), in register Reg(i) of every processor PE(i) the number of processors PE(j) such that $j< i$ and Cond(j) is true [NS].

Number(Reg(i),Cond(i)): Compute, in time O(log N), in register Reg(i) of every processor PE(i) the number of processors PE(j) such that Cond(j) is true.

Concentrate([$Reg_1(i)$,...,$Reg_z(i)$],Cond(i)): This operation includes an initial Rank(R(i), Cond(i)) operation. Then for each PE(i) with Cond(i) = true, registers $Reg_1(i)$,...,$Reg_z(i)$ are copied to PE(R(i)), z=O(1). The time complexity of this operation is also O(log N) [NS].

Route([$Reg_1(i)$,...,$Reg_z(i)$],Dest(i),Cond(i)): Every processor PE(i) has z=O(1) data registers $Reg_1(i)$,...,$Reg_z(i)$, a destination register Dest(i), and a boolean condition register Cond(i). It is assumed that the destinations Dest(i) are monotonic; i.e., if $i<j$ then $Dest(i)<Dest(j)$. This operation routes, for every processor PE(i) with Cond(i) = true, all registers $Reg_1(i)$,...,$Reg_z(i)$ to processor PE(Dest(i)); it can be implemented with an O(log N) time complexity by using a Concentrate operation followed by a Distribute operation described in [NS].

RouteAndCopy([$Reg_1(i)$,...,$Reg_z(i)$],Dest(i),Cond(i)): Under the same assumptions as for the Route operation, this operation routes, for every processor PE(i) with Cond(i) = true, a copy of registers $Reg_1(i)$,...,$Reg_t(i)$ to processors PE(Dest(i - 1) + 1), ..., PE(Dest(i)), each; it can be implemented with an O(log(N)) time complexity by using a Concentrate followed by a Generalize operation described in [NS].

Reverse([$Reg_1(i)$,...,$Reg_z(i)$],Start,End): This operation routes for every PE(i) with Start ≤ i ≤ End, its registers $Reg_1(i)$, ..., $Reg_z(i)$, z=O(1), to PE(Start + End - i); i.e., it reverses the contents of those registers for the sequence of processors between PE(Start) and PE(End). Reversing, in the entire hypercube, a sequence of n values (each stored in one processor) corresponds to routing each value stored at processor PE(i) to processor PE(i'), where i' is obtained from i by inverting all bits in its binary representation. Hence, this operation can be implemented in time log(n) similarly to the Concentrate/Distribute operation described in [NS].

BitonicMerge([$Reg_1(i)$,...,$Reg_z(i)$],Key(i),Left,Peak,Right): This operation is the well known bitonic merge [B]. It converts in time O(log N) a bitonic sequence (with respect to register Key(i)) into a sorted sequence; it simultaneously permutes the registers $Reg_1(i)$,...,$Reg_t(i)$ (z=O(1)). Here, we apply it to a particular bitonic sequence consisting of an increasing sequence starting at PE(Left) and ending at PE(Peak) followed by a decreasing sequence starting at PE(Peak+1) and ending at PE(Right).

BitonicMerge2([$Reg_1(i)$,$Reg_2(i)$,$Reg_3(i)$],$Key_{11}(i)$,$Key_{12}(i)$,$Key_{21}(i)$,$Key_{22}(i)$,$Left_1$,$Right_1$,$Left_2$,$Right_2$): This operation simulates a 2N hypercube. It merges the sorted sequences stored in registers Reg_1 and Reg_2, and stores the resulting sequence into register $Reg_3(1)$, ..., $Reg_3((Right_1-Left_1 + 1) + (Right_2-Left_2 + 1))$. The principal keys for merging are $Key_{11}(i)$ and $Key_{21}(i)$, while the secondary keys are $Key_{12}(i)$ and $Key_{22}(i)$. The first sequence is stored in processors PE($Left_1$) to PE($Right_1$) while the second sequence is stored in PE($Left_2$) to PE($Right_2$).

Sort([$Reg_1(i)$,...,$Reg_z(i)$],Key(i)): This operation refers to $O(\log^2 n)$ time bitonic sort [B] with respect to Key(i); it simultaneously permutes the registers $Reg_1(i)$,...,$Reg_z(i)$ (z=O(1)).

ALGORITHMIC UTILISATION OF LOCAL MEMORY IN MESH CONNECTED COMPUTERS

N. BLAIS and H. ELGINDY*
School of Computer Science,
McGill University
Montréal, Québec, Canada

Abstract

We show that the use of sufficiently large local memory can improve the bounds on the time needed to solve problems on a Mesh Connected Computer (MCC), with or without multiple broadcasting. The main problems discussed here are *semigroup computation* (e.g. find the maximum value in a set) and *selection.* For example we present an algorithm which selects the n^{th} smallest among N items, on a MCC with multiple broadcasting and $N^{6/7}$ processing elements, in $O(N^{1/7}\log N)$ steps. To acheive this, we need that each processor has $O((N^{1/7})$ memory locations. The bounds for selection on a MCC with and without multiple broadcasting are respectively $O(N^{1/6}\log^{2/3} N)$ and $\Theta(\sqrt{N})$ when each of the N processors has a constant number of registers. Further, all our results are extendable to MCC's of any dimensions.

1 INTRODUCTION

Since Kung [7] proposed *systolic architecture* as a parallel processing scheme suitable for VLSI implementation, parallel algorithms for a variety of applications have been designed to run efficiently on special purpose systolic machines [11]. The running time of these algorithms is usually constrained by the processing time within each cell, information flow through the cells, and the time required for moving data over a long distance.

To overcome the latter constraint, *broadcasting* has been proposed as a mechanism for moving data quickly over long distances. A number of algorithms [4,6,9] have demonstrated that the computational time can be substantially improved when broadcasting is incorporated in systolic designs. However, broadcasting cannot solve information flow problems. For example, *sorting* requires $\Omega(\sqrt{N})$ time with or without broadcasting on a mesh connected computer with N processing elements.

---

*Supported in part by NSERC under grant OGP0036737

With today's advances in VLSI technology, it is feasible to provide a large local memory at each processing element. This local memory facilitates the storage of duplicated copies of data to increase the reliability/fault-tolerance of the design. Our objective is to study the use of *data duplication* in improving the computational time of problems constrained by either the need of moving data over a long distance or by large information flow. In a previous paper [2], we have demonstrated the use of data duplication in array architectures by developing an $O((N \log N)^{1/2})$ running time implementation for the *enumeration sorting* method on an enhanced linear array of N processors, each with a memory of size $O((N/\log N)^{1/2})$. In this paper we demonstrate its use in mesh connected architectures by presenting an $O(N^{1/7} \log N)$ time systolic algorithm for selection on N items with $N^{6/7}$ processors that have a local memory of size $O(N^{1/7})$. Our method, which is based on exploiting the ideas of sequential median computation [3], was also used in [5] to develop an optimal speed-up parallel algorithm for triangulating simplicial point sets for the PRAM model of computation.

We describe the specifications of the architecture to be used in section 2. Then we recall previous results and derive agorithms for semigroup computations in section 3. Selection is presented the same way in section 4 and finally we conclude with directions for future research in section 5.

2 ENHANCED MESH CONNECTED COMPUTERS

A k-dimensional Mesh Connected computer (**MCC**) consists of P_M processing elements arranged on a k-dimensional array such that each processor is located at an entry point and connected to its $2k$ nearest neighbors for local data transfer. For the purpose of this paper, we define:

Definition 2.1 *A k-***EMCC** *(*k-dimensional Enhanced Mesh Connected Computer*) is a k-***MCC** *for which each processor has an exclusive access to a local random access memory of size S_P.*

We label each processor with a unique tuple $(i_1, i_2, \ldots, i_k)$ where the i_j's are the coordinates of the processor that vary from 1 to ${P_M}^{1/k}$. We also assume that each processor stores its label in some register and can use it during the execution of our algorithms. From this statement we can define:

Definition 2.2 *A k-***EMCC** *with multiple broadcasting is a k-***EMCC** *for which a bus connects every processor that have a label differing by one and only one coordinate.*

For example, a 2-**EMCC** is made of P_M processors arranged on a ${P_M}^{1/2} \times {P_M}^{1/2}$ square grid and a 2-**EMCC** with multiple broadcasting is a 2-**EMCC** for which one broadcast bus per row and column have been added (see [6]).

Computation in an **EMCC** proceeds in synchronous *cycles*. During a cycle, each processor can perform the following tasks:

1. transfer data to/from its k buses (if any),
2. transfer data to/from its neighbouring processors, and
3. perform a constant number of arithmetic and logical operations.

Furthermore, algorithms designed for a k-**EMCC** must ensure that no two processing elements should attempt to broadcast to the same bus during the same cycle.

The main technique used in algorithms for k-**MCC**'s is based on splitting the mesh structure into *groups* of processors and each group into *blocks*. Groups (resp. blocks) are square submeshes of equal size that form a partition of the mesh (resp. group) i.e. each processor belongs to one and only one group (resp. block). Let S, P, B, G, and M denote respectively the size of the storage, the number of processors, blocks and groups, and the whole mesh. We use the symbol X_Y where X and $Y \in \{S, P, B, G, M\}$ to represent the number of X in Y. For example, S_P represents the size of local memory in each processor, and P_G represensts the number of processors in a group.

3 SEMIGROUP COMPUTATIONS

This section recalls some previous results on solving the semigroup computation problem on k-MCC's. We then present algorithms that use the local memory of the k-EMCC to lower the time upper bounds for this problem on such a machine.

Let $\circ$ be a semigroup operation (e.g. $+$, $-$, $\times$, *maximum*, *minimum*) and $v_1, v_2, \ldots, v_N$ be integer values. We want to compute $v_1 \circ v_2 \circ \ldots \circ v_N$ and sotre the result in some designated in advance processor.

Theorem 3.1 *[9] On a k-*MCC* with N processing elements, the time needed to perform any semigroup operation on N items is $\Theta(N^{1/k})$.*

The next theorem shows how local memory can be used to improve this bound with a smaller number of processors involded.

Theorem 3.2 *On a k-*MCC* with $N^{k/k+1}$ processing elements (each with $N^{1/k+1}$ memory locations), the time needed to perform any semigroup operation on N items is $\Theta(N^{1/k+1})$.*

Proof. We give the algorithm only for $k = 2$ but it can be easily extended to any k.

1. All processors compute in parallel the result of the semigroup operation on the $O(N^{1/3})$ items stored in their local memory
2. Call the algorithm of Theorem 3.1 to perform the semigroup operation on the $O(N^{2/3})$ results of step 1.

Analysis

The running times of steps 1 and 2 are respectively in $O(N^{1/3})$ and $O((N^{2/3})^{1/2})$. This yields a total running time of $O(N^{1/3})$ which is optimal since the product of the number of processors involved and the total running time is $O(N)$.

□

In the standard case (N items, N processors) we subdivide the mesh into $N^{1/k+1}$ groups of $N^{k/k+1}$ processing elements and let each group have a copy of the whole set of N items. Each group then runs in parallel and independantly the algorithm of Theorem 3.2. Naturally, the involvement of all groups in the computation is only justified if all of them have to know the final result. Otherwise some group may stay idle. This remark also holds for k-**EMCC**'s with multiple broadcasting on which we apply the same strategy.

Theorem 3.3 *[6] On a* k-**MCC** *with multiple broadcasting and* N *processing elements the time needed to perform any semigroup computation on* N *items is* $\Theta(N^{1/k(k+1)})$.

When $k = 2$ for example, Theorem 3.2 says that we can find the maximum value among N items with N processors in $\Theta(N^{1/6})$ if we use a multiple broadcasting device. Working with local memory we get:

Theorem 3.4 *On a* k-**EMCC** *with multiple broadcasting and* $N^{k(k+1/k(k+1)+1}$ *processing elements (each with* $N^{1/k(k+1)+1}$ *memory locations), the time needed to perform any semigroup computation on* N *items is* $\Theta(N^{1/k(k+1)+1})$.

Proof. The proof is similar to the one of Theorem 3.2.

□

For example a 2-**EMCC** with multiple broadcasting can solve the problem in $\Theta(N^{1/7})$ time with $N^{6/7}$ processors (each with a storage per processor of $N^{1/7}$).

4 THE n-SELECTION PROBLEM

We present here the speedup obtained for the n-selection problem on our model of computation. This problem consists of finding the n^{th} smallest value in a set of N values. The *median* problem is a special case where $n = \lfloor N/2 \rfloor$.

On a k-**MCC** without any broadcasting the complexity of this problem is the same as sorting i.e. $\Theta(N^{1/k})$. We state:

Theorem 4.1 *On a* k-**EMCC** *with* $N^{k/k+1}$ *processing elements (each with* $N^{1/k+1}$ *memory locations), the time needed for* n*-selection on* N *items is* $O(N^{1/k+1} \log N)$.

Proof. Again, we give the algorithm only for $k = 2$ but it can be easily extended to any k.

1. let all items be initially *unmarked.*
2. repeat the following steps:
   (a) (*processor level computation*) processor p selects the median m_p of unmarked items in its local memory, using the sequential algorithm in [3], and computes the number of unmarked items w_p stored in local memory as their *weights.*
   (b) (*mesh level computation*) processors in the mesh computes the *weighted median* m of the medians found in the processor level computation. This number which is in fact a *pseudo* median of the unmarked items stored in all local memories, is an unmarked item that partitions the set into two subsets whose sizes are a fraction of the total size. We compute it as follows:
      i. sort the records (m_p, w_p) using a mesh sorting algorithm.
      ii. choose i such that

$$\sum_{p=1}^{i-1} w_p < \sum_{p=1}^{P_M} w_p/2 \leq \sum_{p=1}^{i} w_p$$

      and let $m = m_i$.
   (c) based on the selection parameter n and the rank of the weighted median m selected in step 2b, each processor scans its local memory and marks the items that cannot have the rank n among the N items.

   until all items are marked

Analysis

The running time of every step is:

1. $O(S_P)$ time
2. this step iterates $O(\log_{1/\alpha} N)$ times since it eliminates a fraction α $(0 < \alpha < 1)$ each time. Such an iteration takes:
   (a) $O(S_P)$ time
   (b) this step takes $O(S_P + \sqrt{P_M})$ time for its combination of sorting, summing, searching and pipelining operations inside the group
   (c) $O(S_P + \sqrt{P_M})$ for computing, pipelining the rank of m and for local computation

This yields a total running time of $O((S_P + \sqrt{P_M}) \log N)$. Time optimality $(O(N^{1/3} \log N))$ is obtained for this algorithm when we set:

$$S_P = N^{1/3} \text{ and}$$

$$P_M = O(N^{2/3})$$

while the general settings for any k are:

$$S_P = N^{1/k+1} \text{ and}$$

$$P_M = O(N^{k/k+1})$$

for an $O(N^{1/k+1} \log N)$ time.

□

We can get a slight improvement on the running time and the space required by adding a multiple broadcasting device to the k-**EMCC**:

Theorem 4.2 *On a k-**EMCC** with multiple broadcasting and $N^{k(k+1)/k(k+1)+1}$ processing elements (each with at least $N^{1/k(k+1)+1}$ memory locations), the time needed for n-selection on N items is $O(N^{1/k(k+1)+1} \log N)$.*

Proof. The agorithm is almost identical to the one of theorem 4.1 but we use a different strategy to take advantage of the buses. The mesh is partitioned into G_M groups each organnised into B_G blocks.

1. let all items be initially *unmarked.*
2. repeat the following steps:
   (a) (*processor level computation*) processor p selects the median m_p of unmarked items in its local memory using a sequential algorithm and computes the number of unmarked items w_p stored in local memory as their *weights.*
   (b) (*block level computation*) processors in block b computes the *weighted median* m_b of the medians found in the processor level computation. This number which is in fact a *pseudo* median of the unmarked items stored in all local memories, is an unmarked item that partitions the set into two subsets whose sizes are a fraction of the total size. We compute it as follows:
      i. sort the records (m_p, w_p) using a mesh sorting algorithm.
      ii. choose i such that

      $$\sum_{p=1}^{i-1} w_p < \sum_{p=1}^{P_B} w_p/2 \leq \sum_{p=1}^{i} w_p$$

      Let $m_b = m_i$ and $w_b = \sum_{p=1}^{P_B} w_p$ where w_b represents the total number of unmarked items in the block.
   (c) (*group level computation*) processors in each group cooperate in computing the weighted median of the block medians as follows:
      i. *pack* the records (m_b, w_b) just obtained into a square submesh of sizes $\sqrt{B_G} \times \sqrt{B_G}$ inside the group, one record per processor. This could be completed exclusively with the help of row and column buses as long as the number of groups in the mesh is less than processors in a block:

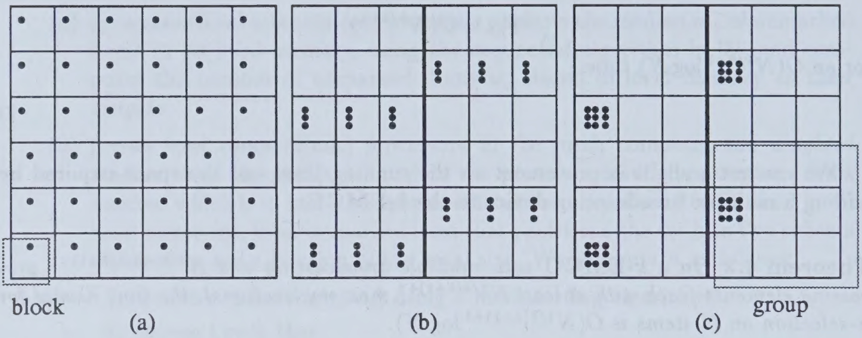


Figure 1: Packing data (represented by dots) resulting from block calculation to a little submesh inside each group. (a) Initial state i.e. each block has stored its median in a definite processor. (b) Column buses pack data vertically. (c) Then row buses pack horizontally.

        - column buses pack the records vertically
        - then row buses pack them horizontally (see figure 1)

    ii. compute the weighted median m_g and its weight w_g of the items just packed

(d) (*mesh level computation*) processors in the whole mesh compute the weighted median of the group medians as follows:

    i. pack the records (m_g, w_g) just obtained into a square submesh of size $\sqrt{G_M} \times \sqrt{G_M}$ inside the mesh, one record per processor. This could be again completed exclusively with the help of row and column since all the buses can participate:

    ii. compute the weighted median m of the items just packed

(e) based on the selection parameter n and the rank of the weighted median m selected in step 2d , each processor scans its local memory and marks the items that cannot have the rank n among the N items.

until all items are marked

Analysis

1. $O(S_P)$ time
2. this step iterates $O(\log_{1/\alpha} N)$ times since it eliminates a fraction α $(0 < \alpha < 1)$ each time. Such an iteration takes:
    (a) processor level is done in $O(S_P)$ time

(b) the block level computation of the median is done in $O(S_P + \sqrt{P_B})$

(c) the group level:

  i. packing data is done in $O(\sqrt{B_G})$

  ii. the median computation is done in $O(S_P + \sqrt{B_G})$ since it is performed in a smaller region of the group

(d) the mesh level:

  i. packing data is done in $O(\sqrt{G_M})$

  ii. the median computation is done in $O(S_P + \sqrt{G_M})$ since it is performed in a smaller region of the mesh

(e) $O(S_P + {P_M}^{1/6})$ for computing, broadcasting the *real* rank of m and for local computation. This is equivalent to a semigroup operation.

This leads to an overall running time of $O((S_P + \sqrt{P_B} + \sqrt{B_G} + \sqrt{G_M}) \log N)$. Time optimality $(O(N^{1/7} \log N))$ is obtained for this algorithm when we set:

$$S_P = N^{1/7} \text{ and}$$

$$P_B = B_G = G_M = O(N^{2/7})$$

The general settings for any k are:

$$S_P = N^{1/k(k+1)+1} \text{ and}$$

$$P_M = O(N^{k(k+1)/k(k+1)+1})$$

for a $O(N^{1/k(k+1)+1} \log N)$ time. Note that for a given k the algorithm will iterate $O(\log\ N)$ times but on $k+1$ levels (processors, blocks, groups, ...).

□

5 CONCLUDING REMARKS

We have presented four algorithms running on a k-EMCC that lowers time bounds for solving two different problems. They achieve their efficiency by striking a balance between local computation, local communication and global communication.

Our future research will follow three main directions. First, the research of better algorithms or lower bounds for selection and sorting on k-EMCC's. Second, the use of local memory in improving the running time of algorithms for other architectures. Third, the use of local memories in the design of more efficient algorithms for scientific computing problems.

References

[1] S. Akl *Parallel Sorting Algorithms*, Academic Press, 1985.

[2] N. Blais, H. ElGindy, D.S. Mahal, and N. Mouawad, "Algorithmic utilization of local memory in linear arrays," *Proceedings of Canadian Conference on Very Large Scale Integration*, Halifax, Nova Scotia, October 1988, pp 69–73.

[3] M. Blum, W.R. Floyd, V.R. Pratt, R.L. Rivest, and R.E. Tarjan, "Time bounds for selection," *J. Computer and System Sciences*, vol 7, 1972, pp 448–461.

[4] S.H. Bokhari, "Finding maximum on an array processor with a global bus," *IEEE Transactions on Computers*, vol C–33, February 1984, pp 133–139.

[5] H. ElGindy, "An optimal speed-up parallel algorithm for triangulating simplicial point sets in space," *International J. of Parallel Programming*, vol 15, October 1986, pp 389–398.

[6] V.K.P. Kumar and C.S. Raghavendra, "Array processor with multiple broadcasting," *J. of Parallel and Distributed Computing*, vol 4, 1987, pp 173–190.

[7] H.-T. Kung, "Let's design algorithms for VLSI systems," *Proceedings of Caltech Conference on VLSI: Architecture, Design, Fabrication*, 1977, pp 65–90.

[8] R. Miller and Q.F. Stout, "Mesh computer algorithms for computational geometry," Technical Report 86-18, Department of Computer Science, State University of New York at Buffalo, Buffalo, New York, July 1986.

[9] Q.F. Stout, "Mesh-connected computers with broadcasting," *IEEE Transaction on Computers*, vol C–32, September 1983, pp 826–830.

[10] C.D. Thompson and H.T. Kung "Sorting on a Mesh-Connected Parallel Computer," *IEEE Transaction on Computers*, vol 20, 1977, pp 263-271.

[11] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984.

CAPACITY FOR A CLASS OF ADDITIVE DIMENSION-LIMITED CHANNELS

CHARLES R. BAKER
Department of Statistics
University of North Carolina
Chapel Hill, N.C. 27599-3260

ABSTRACT

Coding capacity is determined for a class of additive Gaussian channels. The channels may be with or without memory, stationary or nonstationary. The constraint is partially given in terms of an increasing family of finite-dimensional subspaces. A general expression for the capacity is obtained, which also depends upon the relation between the noise covariance and the constraint on the generalized signal-to-noise energy ratio for the code words.

INTRODUCTION

How does one determine coding capacity for the discrete-time memoryless nonstationary Gaussian channel when the noise variances have several limit points?

For a continuous-time Gaussian channel of fixed time duration, is there a reasonable method whereby one can define and compute a "channel capacity"?

This paper provides answers to these questions. Attention is focused on channels where the allowable dimensionality of the code word set is a key component of the problem. As such, the discrete-time Gaussian channel is a special case. A general result for coding capacity is given. This result applies to all discrete-time Gaussian channels: with or without memory, stationary or non-stationary. It also provides a method for defining capacity of a continuous-time Gaussian channel of fixed time duration. Examples are given for discrete-time and continuous-time Gaussian channels.

Upper bounds on coding capacity using this approach have also been obtained for additive nonGaussian channels [3]. Proofs of the results given here are omitted; see [3] or [4].

FRAMEWORK

In the next few sections, the setting and definitions for the coding capacity problem will be given. In order to illustrate these concepts and definitions, the classical discrete-time channel will be frequently employed.

It is assumed that the noise sample paths belong to a real separable Hilbert space H, where H has inner product $\langle \cdot , \cdot \rangle$ and associated norm $\| \cdot \|$.

The noise is described by a set function μ_N. μ_N will be a finitely-additive Gaussian probability defined on the cylinder sets of H: the collection of all sets of the form $\{x: (\langle x,u_1\rangle,\dots,\langle x,u_n\rangle) \in D_n\}$, where $n \geq 1$, D_n is a Borel set in $\mathbb{R}^n$, and $u_1,\dots,u_n$ are any n elements of H. Thus, if H_0 is any finite-dimensional subspace of H, and P_0 is the projection operator in H having range H_0, let μ_0 be defined on the Borel sets of H_0 by $\mu_0(A) = \mu_N\{x: P_0x \in A\}$. μ_0 is then a countably-additive Gaussian probability. Our basic assumptions on the noise are that

a) $\int_H \langle x,y\rangle^2 d\mu_N(x) < \infty$ all y in H, and

b) $\int_H \langle x,y\rangle d\mu_N(x) = 0$ for all y in H.

Assumption (a) means that μ_N has a covariance operator R_N, which is linear, bounded, and non-negative, and also implies that the noise mean exists. Assumption (b) is that the noise has zero mean. We can assume WLOG that R_N is strictly positive on H. R_N is defined as $\langle R_Nu,v\rangle = \int_H \langle u,x\rangle\langle v,x\rangle d\mu_N(x)$ for u,v in H.

μ_N Gaussian means that for any $y \in H$ the distribution function P_y is Gaussian, $P_y(\alpha) = \mu_N\{x: \langle x,y\rangle \leq \alpha\}$.

CONSTRAINTS

Let R_W be a strictly positive covariance operator in H satisfying $\text{range}(R_W^{1/2}) \subset \text{range}(R_N^{1/2})$.

(H_n), $n \geq 1$, will denote a family of finite-dimensional subspaces of H, such that for all $n \geq 1$,

a) $H_n \subset H_{n+1}$

b) $\dim(H_n) = n$.

Let $\tilde{P}_n$ be the projection operator with range equal to H_n, and define $R_{W,n} = \tilde{P}_n R_W \tilde{P}_n$. $R_{W,n}$ is then strictly positive on H_n. For x in H_n, the norm $\|x\|_{W,n}$ is defined by $\|x\|_{W,n} = \|y\|$, where y is the unique element of H_n that satisfies $R_{W,n}^{1/2}y = x$. Formally, we write $\|x\|_{W,n} = \|R_{W,n}^{-1/2}x\|$ for x in H_n; although $R_{W,n}^{-1/2}$ does not exist on H, it is well-defined on H_n.

The constraints on the code words are now as follows: For each $n \geq 1$, the admissible code words $x_1,\dots,x_{K(n)}$ belong to H_n and satisfy $\|x_i\|_{W,n}^2 \leq nP$ for $i = 1,2,\dots,K(n)$.

As an important example of such constraints, consider the classical discrete-time memoryless channel with a simple energy constraint. This can be formulated in the above terms by taking $H = \ell_2$ and R_W the identity, giving $H_n = \{x \text{ in } \ell_2 \colon \; x_i = 0 \text{ for } i > n\}$

$$\|x\|^2_{W,n} = \sum_{i=1}^{n} x_i^2 \qquad \text{for } x \text{ in } H_n .$$

Another example involves the continuous-time channel with fixed transmission time, T. In this case, H can be taken as $L_2[0,T]$, and H_n as the linear span of $\{e_1, e_2, \ldots, e_n\}$, where $\{e_n, \; n \geq 1\}$ is an infinite orthonormal set in $L_2[0,T]$.

DEFINITION OF CODING CAPACITY

Let $(H_n, \|\cdot\|_{W,n})$ define the constraints. Let μ_N be the noise probability on H, and let μ_N^n be the probability on H_n induced from μ_N by the projection operator $\tilde{P}_n$: $\mu_N^n[A] = \mu_N\{x \colon \tilde{P}_n x \in A\}$, for A any Borel set in H_n. It is not required that μ_N be countably additive; thus, for example, the discrete-time memoryless stationary Gaussian channel is included in this formulation.

For fixed $n \geq 1$, a *code* (k, n, ϵ) [1] is a set of k code words $\{x_1, \ldots, x_k\}$ and k disjoint Borel sets in H_n such that the elements of $\{x_1, \ldots, x_k\}$ obey the constraints, and

$$\mu_N\{y \colon \tilde{P}_n y + x_i \in C_i\} \geq 1 - \epsilon \text{ for } i = 1,2,\ldots,k.$$

A real number $R \geq 0$ is an admissible rate if there exists an infinite sequence of codes $([e^{n_i R}], n_i, \epsilon_{n_i})$ with $\epsilon_{n_i} \to 0$ as $i \to \infty$, where [r] is the integer part of the real number r.

The coding capacity is then the supremum of the set of admissible rates.

As can be seen, this definition of coding capacity contains that of the classical discrete-time channel as a special case, defining the constraint family $(H_n, \|\cdot\|_{W,n})$ as in the previous section. More generally, the capacity gives an indication of the effect on size of the code word set that can be obtained by increasing the dimensionality of the code word set, while requiring that $\liminf_{n \to \infty} \epsilon_n = 0$.

The constraint on the transmitted signal is given in terms of a covariance operator R_W in H. A basic assumption is that range($R_N^{1/2}$) contains range($R_W^{1/2}$). The existence of such an operator, and the assumption on range relations, are necessary in order that the information capacity be finite [2]; it will be seen that finite information capacity is necessary in order to have finite coding capacity. Thus, the formulation of the problem is quite general (so long as an average-type constraint is to be used). Under this assumption, there exists a self-adjoint operator S in H such that $R_N = R_W^{1/2}(I+S)R_W^{1/2}$, where $(I+S)^{-1}$ exists and is bounded (see [2, Prop. 1] for ramifications of this fact). The limit points of the spectrum of S play a key role in the analysis [3], [4]. These limit points (the essential spectrum of S) consist of all eigenvalues of infinite multiplicity, all limits of sequences of distinct eigenvalues, and all points of the continuous spectrum [8, p. 363]. "Essential spectrum" is the modern terminology for this set; it will be denoted by $\sigma_{ess}(S)$. The continuous spectrum of S consists of all real numbers λ such that the range of $(S-\lambda I)$ is not closed.

In many applications, the constraint will be given by a time-invariant linear filter f with transfer (frequency) function $\hat{f}$. In such cases, $|\hat{f}|^2$ defines a spectral density and thereby the operator R_W.

EXPRESSIONS FOR EVALUATION OF THE CODING CAPACITY

The noise has probability μ_N and covariance operator R_N. μ_N^n is the noise probability on H_n, defined by $\mu_N^n(C) = \mu_N\{x\colon \tilde{P}_n x \in C\}$ for C a Borel set in H_n. R_W is the covariance operator defining the constraint on the code words. $R_{W,n} = \tilde{P}_n R_W \tilde{P}_n$ and $R_{N,n} = \tilde{P}_n R_N \tilde{P}_n$. Let I_n be the identity in H_n; let S_n be the self-adjoint operator mapping H_n into H_n defined by $R_{N,n} = R_{W,n}^{1/2}(I_n+S_n)R_{W,n}^{1/2}$. $\|S_n x\| \equiv 0$ if x is orthogonal to H_n. Since S_n is self-adjoint as an operator in H_n, it has n orthonormal eigenvectors belonging to H_n with corresponding eigenvalues $\beta_1^n \le \beta_2^n \le \dots \le \beta_n^n$.

Consider now the finite-dimensional channel defined as follows. The additive noise has probability μ_N^n. The input to the channel is described by a probability μ_X on H_n, satisfying $E_{\mu_X}\|x\|_{W,n}^2 \le nP$. Let $C_W^n(nP)$ be the information capacity of this channel. The following well-known result is fundamental to our results.

Lemma 1 [7].

$$C_W^n(nP) = \tfrac{1}{2}\sum_{i=1}^{N(n)} \log\left[\frac{B(n)+1}{\beta_i^n+1}\right]$$

where $\beta_1^n \le \beta_2^n \le \dots \le \beta_n^n$ are the eigenvalues of S_n, $N(n) = \sup\{i \le n:$

$\beta_i^n \leq B(n)\}$, and B(n) is defined by

$$P = \frac{1}{n} \sum_{i=1}^{N(n)} (B(n) - \beta_i^n).$$

PRELIMINARY RESULTS

Our program is to first obtain expressions for $\overline{\lim}_n \frac{1}{n} C_W^n(nP)$. This will then be followed by the result that the coding capacity is equal to $\overline{\lim}_n \frac{1}{n} C_W^n(nP)$. Among the difficulties in evaluation of $\overline{\lim}_n \frac{1}{n} C_W^n(nP)$ is that for each value of n one obtains an expression in terms of the eigenvalues of the operator S_n. The range of S_n is contained in H_n; S_n always has a complete orthonormal set of eigenvectors, and its range space has dimension $\leq n$. Moreover, the eigenvalues of S_n need not be contained in the set of eigenvalues of S_{n+1}; in fact, the eigenvalues of S_{n+1} may not include a single eigenvalue of S_n.

In [4], it is shown that $S_n = V_n S V_n^*$, where V_n is a partially isometric operator. V_n is isometric on range($R_W^{1/2}\tilde{P}_n$), zero on $[\text{range}(R_W^{1/2}\tilde{P}_n]^{\perp}$, and its range is equal to H_n. Let $H_{W_n} \equiv \text{range}(R_W^{1/2}\tilde{P}_n)$. The eigenvalues (β_i^n) of S_n and their multiplicity are then the same as those of the operator $P_{W_n} S P_{W_n}$, where P_{W_n} is the projection operator with range equal to H_{W_n} [4].

$\{e_i,\ i \geq 1\}$ will be used to denote an o.n. set in H such that $H_n = \text{span}\{e_1, \ldots, e_n\}$. Similarly, $\{u_i,\ i \geq 1\}$ will denote an o.n. set such that $H_{Wn} = \text{span}\{u_1, \ldots, u_n\}$. Note that one of these sets is complete for H if and only if the other is complete; completeness is equivalent to $\|\tilde{P}_n x - x\| \to 0$ for all x in H.

CODING CAPACITY

The following theorems give a general result for coding capacity. See [4] for the proof of Theorem 1 and the Corollary.

Theorem 1.

$$(1) \quad \overline{\lim}_{n\to\infty} \frac{1}{n} C_W^n(nP) = \overline{\lim}_{n\to\infty} \tfrac{1}{2} \int_{-1}^{B_n} \log\left[\frac{B_n+1}{\lambda+1}\right] dF_n(\lambda)$$

where B_n is defined by $\quad P = \int_{-1}^{B_n} (B_n - \lambda)\, dF_n(\lambda)$

and $$F_n(\lambda) = \frac{1}{n}\{\#\text{ eigenvalues of } S_n \le \lambda\}.$$

(2) If $\overline{\lim}_n \frac{1}{n} C_W^n(nP) > 0$, and $\lim_{n\to\infty} \frac{1}{n}\{\#\text{ eigenvalues of } S_n < \lambda\}$ exists for all λ in $\mathbb{R}$, then

$$\overline{\lim}_n \frac{1}{n} C_W^n(nP) = \tfrac{1}{2} \int_{-1}^{B} \log[\frac{1+B}{1+\lambda}]dF(\lambda)$$

where F is a distribution function defined by

$$F(\lambda) = \lim_{n\to\infty} \frac{1}{n}\{\#\text{ eigenvalues of } S_n \le \lambda\} = \lim_{n\to\infty} F_n(\lambda),$$

and the constant B is defined by

$$P = \int_{-1}^{B}[B-\lambda]dF(\lambda).$$

(3) $\overline{\lim}_n \frac{1}{n} C_W^n(nP) = 0$ if and only if $\overline{\lim}_{n\to\infty} \frac{1}{n}\{\#\text{ eigenvalues of } S_n < \lambda\} = 0$ for all λ in $\mathbb{R}$. This requires that S be unbounded and occurs in particular if $+\infty$ is the only limit point of $\sigma(S)$.

An alternative form of Theorem 1 can be given, as follows.

Theorem 1A.

Suppose that B is the largest number such that $P \ge \overline{\lim}_n \int_{-1}^{B} [B_n-\lambda]dF_n(\lambda)$.

Then

$$\overline{\lim}_n \frac{1}{n} C_W^n(nP) = \overline{\lim}_n \int_{-1}^{B} \log\left[\frac{B_n+1}{\lambda+1}\right]dF_n(\lambda)$$

where (F_n) and (B_n) are defined as in Theorem 1. If no such B exists, then $\overline{\lim}_n \frac{1}{n} C_W^n(nP) = 0$.

The following result, together with Theorem 1 (and 1A), gives the coding capacity. Theorem 1 enables a proof to be given [3] based upon results due to Ebert [6] and Gallager [7]. This approach also provides error bounds [3].

Theorem 2.

Let $C_W^\infty(P)$ be the coding capacity. $C_W^\infty(P) = \overline{\lim}_n \frac{1}{n} C_W^n(nP)$; when $C_W^\infty(P) = 0$, then the maximum decoding error probability is equal to one for any rate > 0 (i.e., $\liminf_n \epsilon_n = 1$).

APPLICATIONS: DISCRETE-TIME CHANNELS

Prop. 1.

Let $H = \ell_2$, $H_n = \{x\colon x_i = 0 \text{ for } i \geq n\}$, $R_W = I$, and $R_N = \operatorname{diag}[\sigma_i, i \geq 1]$. Suppose that μ_N is Gaussian and that (σ_i) has the limit points $\theta_1 < \theta_2 < \ldots < \theta_K$. Then

$$C_W^\infty(P) = \overline{\lim_n} \sum_{j=1}^{J} \gamma_j^n \log\left[\frac{B_n}{\theta_j}\right]$$

where $\gamma_j^n = M_j^n/n$, M_j^n is the number of elements in the sequence $(\sigma_i, i \leq n)$ belonging to $(\theta_j-\epsilon, \theta_j+\epsilon)$, $\epsilon > 0$ satisfies $2\epsilon < \min\{\theta_{j+1}-\theta_j\colon j \geq 0, \theta_0 \equiv 0\}$, (B_n) is defined by $P = \Sigma_{j=1}^J \gamma_j^n(B_n - \theta_j)$, and J is the largest integer $\leq K$ such that

$$P \geq \overline{\lim_n} \sum_{i \leq J} \gamma_i^n(\theta_J - \theta_i).$$

If $\lim_{n\to\infty} \gamma_i^n \equiv \gamma_i$ exists for $i \leq J$ with J as defined below, then

$$C_W^\infty(P) = \tfrac{1}{2} \sum_{i=1}^{J} \gamma_i \log\left[\frac{B}{\theta_i}\right]$$

where J and B are defined by $P = \Sigma_{j=1}^J \gamma_j(B - \theta_j)$, with θ_J the largest element of $\{\theta_1,\ldots,\theta_K\}$ satisfying $P \geq \Sigma_{i=1}^K \gamma_i(\theta_J - \theta_i)$.

It can be seen that the best choice of (H_n) from the viewpoint of maximizing capacity will be the natural choice $H_n = \{x\colon x_i = 0, i > n\}$ only in special situations. If $\theta_1 < \theta_2 < \ldots < \theta_K$ are the limit points of the noise variances, and μ_N is Gaussian, then an optimum choice of (H_n) is given by $H_n = A_n \cap B$, where

$A_n = \{x\colon x_i = 0, i > k(n)\}$,

$B = \{x\colon x_i = 0, |\sigma_i - \theta_1| \geq \epsilon\}$,

$k(n)$ = smallest integer such that $|\sigma_i - \theta_1| < \epsilon$ for exactly n values of $i \leq k(n)$.

and $\epsilon < \theta_2 - \theta_1$.

This definition of (H_n) gives

$$C_W^\infty(P) = ½ \log[1 + P/\theta_1],$$

which is the maximum possible value for a given S. Of course, this squares perfectly with intuition; the original channel is transformed into a "channel" having limiting noise variance θ_1, at the expense of increasing the transmission time required to achieve a specified decoding error.

The choice of $H_n = \{x: x_i = 0, i > n\}$ gives the optimum (H_n) when μ_N is Gaussian if and only if $\limsup_n \gamma_1^n = 1$. Thus, if choice of (H_n) is part of the system design, then it is only in this case that the capacity is equal to that which is obtained in the classical Gaussian channel. Conversely, the classical channel gives the worst possible choice of (H_n) if and only if $\overline{\lim_n}\, \gamma_i^n = 0$ for all $i \leq K-1$.

APPLICATIONS: CONTINUOUS-TIME CHANNELS OF FIXED DURATION

In this section the code words and noise paths are elements of $L_2[0,T]$, where $T < \infty$ is fixed. The available energy per transmitted code word is P_0: for each fixed n and a given (H_n) and R_W, one has $\|x\|^2_{W,n} \leq P_0$ for each code word x. The question is whether or not arbitrarily small maximum decoding error probability can be achieved by making P_0 arbitrarily large without limiting the cardinality of the code word set. It can be assumed that limiting the cardinality of the code word set is equivalent to limiting its dimensionality.

A complete set of results on capacity can be given for all stationary Gaussian channels where R_N and R_W are defined by rational spectral densities.

Prop. 2.

Suppose that $H = L_2[0,T]$. Let (N_t) be stationary and Gaussian with rational spectral density Φ_N, and suppose that R_W is defined by a rational spectral density Φ_W. Then, for any choice of (H_n):

a) the coding capacity will be non-zero if and only if

$$\lim_{|\lambda|\to\infty} \frac{\Phi_W(\lambda)}{\Phi_N(\lambda)} \equiv \alpha > 0;$$

b) the capacity is given by $C_W^\infty(P) = \frac{1}{2} \log[1 + \alpha P]$.

From (a) of Prop. 2, $C_W^\infty(P) = 0$ if Φ_W/Φ_N is integrable. This is the class of channels considered in the Holsinger-Gallager result for the classical continuous-time channel [7, Sec. 8.5], $T \to \infty$. It may be judged only natural that $C_W^\infty(P)$ is the capacity for the channel restricted to the interval $[0,T]$ and with the constraint $\underset{\sim}{E}\|X\|^2_{W,T} \leq PT$. Since the dimensionality is not constrained in computing $C_W^T(TP)$, $C_W^T(TP \geq C_W^n(nP)$ when $T = n$. Moreover, when T is fixed, the capacity is determined in terms of $(\frac{1}{n} C_W^n(nP))$. However, if R_W and R_N are given by rational spectral densities such that $\lim\limits_{|\lambda|\to\infty} \frac{\Phi_W(\lambda)}{\Phi_N(\lambda)}$ is finite and non-zero, then the capacity is finite and non-zero for both the classical channel [5] and (by Prop. 2) the fixed-time channel.

Additional results for both continuous-time and discrete-time Gaussian and nonGaussian channels are contained in [3].

REFERENCES

[1] R.B. Ash, *Information Theory*, Interscience, New York, 1965.

[2] C.R. Baker, Capacity of the mismatched Gaussian channel, *IEEE Trans. on Inform. Theory*, IT-33, 802-812 (1987).

[3] C.R. Baker, Coding capacity for a class of additive channels, to appear.

[4] C.R. Baker, Information capacity of dimension-limited channels, to appear.

[5] C.R. Baker and S. Ihara, Information capacity of the stationary Gaussian channel (to appear).

[6] P.M. Ebert, Error bounds for parallel communication channels, *MIT Research Lab. of Electronics*, Tech. Rept. 448 (1966).

[7] R.G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York (1968).

[8] F. Riesz and B. Sz-Nagy, *Functional Analysis*, Ungar, New York, 1955.

This research was supported by ONR Contract N00014-86-K-0039, ONR Grant N00014-89-J-1175 and NSF Grant NCR8713726.

TOTAL CAPACITY OF THE RMS BANDLIMITED K-USER PAM SYNCHRONOUS CHANNEL

ROGER S. CHENG & SERGIO VERDÚ †
Department of Electrical Engineering
Princeton University, Princeton, NJ 08544

ABSTRACT

Continuous-time additive white Gaussian noise channels with strictly time-limited and root mean square (RMS) bandlimited inputs are studied. RMS bandwidth is equal to the normalized second moment of the spectrum, which has proved to be a useful and analytically tractable measure of the bandwidth of strictly time-limited waveforms.

We find the Total Capacity (TC) of the K-user channel under total power and power-weighted average RMS bandwidth constraints. A lower bound to the TC under equal-power constraint is obtained. Total Capacity Ratio (TCR) is defined as the ratio of the K-user TC to K times the single-user capacity. Power (Bandwidth) efficiency is defined as the ratio of the effective power (bandwidth) to the actual power (bandwidth). The effective power (bandwidth) is the corresponding power (bandwidth) needed for a single user channel to achieve the same capacity. We find lower bounds to the TCR and efficiencies which indicate that savings in bandwidth compared to the FDMA scheme can be achieved by the CDMA scheme at the expense of more complicated decoding hardware.

1. INTRODUCTION

In this paper, we deal with the continuous-time Pulse Amplitude Modulation (PAM) Gaussian multiple-access channel (MAC). Each user is assigned a fixed deterministic continuous-time signature waveform, $s_k(t)$, which is time-limited to $[0, T]$ and is modulated linearly by the information stream. Assuming that the transmitters are symbol-synchronous, the channel can be expressed as

$$y(t) = \sum_{i=1}^{n} \sum_{k=1}^{K} b_k(i) s_k(t - iT) + n(t) \tag{1}$$

where $n(t)$ is white Gaussian noise with spectral density, $\frac{N_0}{2}$ and $\{b_k(i)\}$ is the symbol stream transmitted by the k^{th} user.

The capacity region of this channel has been found by Verdú [1] [2]. Denoting $\mathbf{W}$ and $\mathbf{H}$ as the diagonal matrix with the users' powers as its diagonal entries, and the cross-correlation matrix of the normalized signature waveforms, respectively, the capacity region is expressed as

$$\mathbf{C}_V = \left\{ (R_1, R_2, \ldots, R_K) : \sum_{j \in J} R_j \le \frac{1}{2T} \log[\det(\mathbf{I}_{|J|} + \frac{2T}{N_0} \mathbf{W}_J \mathbf{H}_J)] \quad \forall J \subset \{1, \ldots, K\} \right\} \tag{2}$$

---

† This work was supported by the office of Naval Research under Contract N00014-87-K-0054 and by the National Science Foundation under PYI Grant ECSE-8857689

where $\mathbf{A}_J$ is the $|J|\text{x}|J|$ matrix formed by the j^{th} row and column of $\mathbf{A}$ for all $j \in J$. It is clear that, without other constraints, the capacity region is maximized by orthogonal signature waveforms. However, under bandwidth constraints, orthogonal signature waveforms are not necessarily optimal since orthogonality can only be achieved by lowering the symbol rate, $1/T$.

There are many different bandwidth definitions [3]. In this paper, we concentrate on the root mean square (RMS) bandwidth because it is analytically tractable and can be applied to strictly time-limited signals. The RMS bandwidth was introduced by Gabor [4] and studied subsequently in [3], [5] and [6]. It is the square root of the second moment of the energy spectral density ($|S_k(f)|^2$) of the normalized signal which is proportional to the square root of the energy of its derivative.

In the two-user case, the capacity region of the RMS bandlimited PAM channel has been found in [7] and the total capacity (the maximum rate sum over the capacity region) is larger than the single-user capacity with the power equal to the sum of the users' powers. The gain in the total capacity from the single-user to the two-user case can be explained by the increase in the dimensionality of the signal set. We can consider the transmitted signal in a symbol interval as a signal drawn from a signal set. Then, the signal set in the single-user and the two-user case are one-dimensional and two-dimensional, respectively. From this viewpoint, it is easy to see that the total capacity increases as the number of users increases while the total power remains constant.

In this paper, we find the total capacity (TC) of the K-user channel under the total power constraint

$$\operatorname{tr}(\mathbf{W}) \leq W \tag{3}$$

and the power-weighted averaged RMS bandwidth constraint

$$\frac{1}{\operatorname{tr}(\mathbf{W})} \sum_{k=1}^{K} \mathbf{W}_{kk} \int_{-\infty}^{\infty} f^2 \, |S_k(f)|^2 \, df \leq B^2 \tag{4}$$

The power constraint is placed on the total power instead of the individual power since the later requires finding all possible sets of eigenvalues of a positive definite matrix with fixed diagonal entries which is, in general, intractable. The bandwidth constraint is justified because the power-weighted average RMS bandwidth is the RMS bandwidth of the power spectral density of the transmitted signal.

Several performance measures, Total Capacity Ratio, Power efficiency and Bandwidth efficiency, are defined and analyzed. Bounds and limiting values of these measures are also obtained.

2. TOTAL CAPACITY

Theorem 2.1.

The Total Capacity of the K-user RMS bandlimited PAM Gaussian MAC with total power and power-weighted average RMS bandwidth constraints is

$$\mathrm{TC}(B, K, \Lambda) = \max \left\{ \frac{B}{\gamma} \sum_{n=1}^{N} \log[1 + h_n(\lambda)] \right\} \log e \tag{5}$$

where the maximization is over $1 \leq N \leq K$, $\frac{\gamma K \Lambda}{N} \leq \lambda$, *and* $1 \leq \gamma \leq \sqrt{f_N}$ *such that* $\gamma = \sqrt{f_N}$ *iff* $\lambda = \frac{\gamma K \Lambda}{N}$, *and*

$$\sum_{n=1}^{N} h_n(\lambda) = \gamma K \Lambda \tag{6}$$

where

$$h_n(\lambda) \overset{\triangle}{=} \frac{N\lambda f_N + \gamma K \Lambda[\lambda(\gamma^2 - 1) - 1] - n^2(N\lambda - \gamma K \Lambda)}{N(f_N - 1 - \lambda) + \gamma^3 K \Lambda + n^2(N\lambda - \gamma K \Lambda)} > 0 \tag{7}$$

$$f_N \overset{\triangle}{=} \frac{1}{N}\sum_{n=1}^{N} n^2 = \frac{1}{6}(N+1)(2N+1) \tag{8}$$

and the average signal-to-noise ratio is denoted by

$$\Lambda \overset{\triangle}{=} \frac{W}{KN_0B} \tag{9}$$

Proof.

Since the signature waveforms are RMS bandlimited, and the set $\{\phi_i(t,T)\}_{i=1}^{\infty}$ where

$$\phi_i(t,T) \overset{\triangle}{=} \begin{cases} \sqrt{\frac{2}{T}}\sin(\frac{\pi i t}{T}) & \text{if } t \in [0,T]; \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

forms a complete orthonormal basis for all RMS bandlimited signals, we can write

$$\begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_K(t) \end{bmatrix} = \mathbf{A}^T \boldsymbol{\Phi}(t,T) \tag{11}$$

where $\boldsymbol{\Phi}(t,T) \overset{\triangle}{=} [\phi_1(t,T)\ \phi_2(t,T)\ \ldots]^T$. Then, the power-weighted average RMS bandwidth constraint can be written, via the Parsaval's theorem, as

$$\begin{aligned} \frac{1}{\mathrm{tr}(\mathbf{W})}\sum_{k=1}^{K} \mathbf{W}_{kk}\int_0^T [\frac{d}{dt}s_k(t)]^2\, dt &= \frac{1}{(2T)^2\mathrm{tr}(\mathbf{W})}\mathrm{tr}(\mathbf{W}\mathbf{A}^T\boldsymbol{\Pi}\mathbf{A}) \\ &= \frac{1}{(2T)^2\mathrm{tr}(\mathbf{W})}\mathrm{tr}(\boldsymbol{\Pi}\mathbf{A}\mathbf{W}\mathbf{A}^T) \\ &\leq B^2 \end{aligned} \tag{12}$$

From the capacity region in (2), it is clear that the total capacity is maximized when $\mathrm{tr}(\mathbf{W}) = W$. We denote the time-bandwidth product by $\gamma \overset{\triangle}{=} 2BT$, the average signal-to-noise ratio by $\Lambda \overset{\triangle}{=} \frac{W}{KN_0B}$, and the eigenvalues of $\frac{2T}{N_0}\mathbf{W}\mathbf{H}$ by λ_k such that $\lambda_j \leq \lambda_i, \forall i < j \leq K$. Then, the total capacity becomes

$$\mathrm{TC_V} = \frac{B}{\gamma}\sum_{k=1}^{K}\log[1 + \lambda_k] \tag{13}$$

and the power constraint becomes

$$\sum_{k=1}^{K}\lambda_k = \gamma K \Lambda \tag{14}$$

Since the eigenvalues of $\frac{2T}{N_0}\mathbf{WH}$ are also the eigenvalues of $\frac{2T}{N_0}\mathbf{AWA}^T$, and once $\{\lambda_k\}_{k=1}^K$ are fixed, the left hand side of (12) is minimized when $\mathbf{AWA}^T$ is diagonal with decreasing diagonal entries, we can rewrite (12) as

$$\sum_{k=1}^{K} k^2\lambda_k \le \gamma^3 K\Lambda \tag{15}$$

For fixed T, the total capacity is found by maximizing (13) over all $\lambda_k \ge 0$, $k = 1, \dots, K$ under the constraints (14) and (15). Using the Kuhn-Tucker Theorem, we form the Lagrangian

$$-\sum_{k=1}^{K} \log[1+\lambda_k] + x(\sum_{k=1}^{K} \lambda_k - \gamma K\Lambda) + y(\sum_{k=1}^{K} k^2\lambda_k - \gamma^3 K\Lambda) \tag{16}$$

and obtain the necessary conditions:

$$\lambda_n \;=\; \frac{1}{x+yn^2} - 1 \;>\; 0 \qquad n = 1, \dots, N. \tag{17}$$

and $\lambda_n = 0$ for all $n > N$,

$$y(\sum_{n=1}^{N} n^2\lambda_n - \gamma^3 K\Lambda) = 0 \tag{18}$$

and $0 \le y$.

Rewriting (17) as $(x+yn^2)(1+\lambda_n) = 1$, and summing over all n, we have, from (14) and (18),

$$(N+\gamma K\Lambda)x + (Nf_N + \gamma^3 K\Lambda)y = N \tag{19}$$

Particularizing (17) to $n = 1$, and substituting in (19), we have

$$y = \frac{N\lambda_1 - \gamma K\Lambda}{(1+\lambda_1)(N(f_N-1) + \gamma K\Lambda(\gamma^2-1))} \tag{20}$$

and

$$x = \frac{N(f_N - 1 - \lambda_1) + \gamma^3 K\Lambda}{(1+\lambda_1)(N(f_N-1) + \gamma K\Lambda(\gamma^2-1))} \tag{21}$$

Substituting (20) and (21) into (17), and denoting λ_1 by λ and λ_n by $h_n(\lambda)$, we have (7), and the power constraint in (14) becomes (6).

When $y = 0$, $\lambda = h_n(\lambda) = \frac{\gamma K\Lambda}{N}$ for all $n = 1, \dots, N$. Upon substituting into (15), we have $\sqrt{f_N} \le \gamma$. Since the total capacity becomes $\frac{BN}{\gamma}\log[1+\frac{\gamma K\Lambda}{N}]$ which is monotonically decreasing in γ, the optimal γ is equal to $\sqrt{f_N}$ and (15) is satisfied with equality. If we rewrite (7) and sum up over all n, we have

$$(N\lambda_1 - \gamma K\Lambda)(\sum_{n=1}^{N} n^2 h_n(\lambda) - \gamma^3 K\Lambda) = 0 \tag{22}$$

When $0 < y$, $\frac{\gamma K\Lambda}{N} < \lambda$, and from (22), (15) is again satisfied with equality. Therefore, if we require $\gamma = \sqrt{f_N}$ iff $\lambda = \frac{\gamma K\Lambda}{N}$, (15) is superfluous. Finally, specifying the range of γ and λ and the condition that $\gamma = \sqrt{f_N}$ iff $\lambda = \frac{\gamma K\Lambda}{N}$, we have the desired result. ∎

This theorem gives the exact calculation needed for the TC. The main reason why we cannot obtain a simpler solution is the lack of a closed form expression of $\sum_{n=1}^{N}\frac{1}{a+bn^2}$. However, despite the complicated expression, the TC can be computed once the average signal-to-noise ratio, Λ, and K is given. In Figure 1, we show the TC with different values of K and Λ.

For a given $\mathbf{W}$, we show that any set of signature waveforms, with $\mathbf{A}$ such that $\mathbf{AWA}^T$ is a diagonal matrix with the n^{th} diagonal entry equal to $h_n(\lambda)$, is optimal. However, such an $\mathbf{A}$ does not always exist for any arbitarily given $\mathbf{W}$. For fixed total power, W, finding the set of $\mathbf{W}$ where $\mathbf{A}$ exists is equivalent to finding the possible set of diagonal entries of a positive definite matrix with fixed eigenvalues, which seems intractable. Reversing the problem, one may want to fix the $\mathbf{W}$ and find the total capacity. In general, this is equivalent to finding the possible set of eigenvalues of a positive definite matrix with fixed diagonal entries, which is again intractable.

In the following theorem, we give a lower bound to the TC in the equal-power case where $\mathbf{W} = \frac{W}{K}\mathbf{I}$. Clearly, this is also a lower bound to the capacity of the original channel with the total power constraint in Theorem 2.1.

Theorem 2.2.

The lower bound to the Total Capacity when the users' powers are the same is

$$\mathrm{TC}_{\mathrm{EP}}(B, K, \Lambda) \geq \max_{1 \leq \gamma \leq \sqrt{f_K}} \frac{B}{\gamma} \log\left\{ [1 + \gamma\Lambda \frac{(\gamma^2 - 1) + K(f_K - \gamma^2)}{f_K - 1}][1 + \gamma\Lambda \frac{\gamma^2 - 1}{f_K - 1}]^{K-1} \right\} \tag{23}$$

Proof.

The lower bound is found by exhibiting a symmetric positive definite matrix $\mathbf{H}$, such that the total capacity for that particular signature waveform set is easy to find. We let $\mathbf{H}$ be

$$\mathbf{H} = \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho \\ \rho & \cdots & \rho & 1 \end{bmatrix} \tag{24}$$

the eigenvalues of $\mathbf{H}$ with $0 \leq \rho \leq 1$ to be be specified in the sequel are $1 + (K-1)\rho$ and $1 - \rho$ with multiplicity $K - 1$. Then, the total capacity under the equal-power constraint becomes

$$\begin{aligned} \mathrm{TC}_{\mathrm{V}} &= \frac{B}{\gamma} \log[\det(\mathbf{I}_K + \gamma\Lambda\mathbf{H})] \\ &= \frac{B}{\gamma} \log\left\{[1 + \gamma\Lambda(1 + (K-1)\rho)][1 + \gamma\Lambda(1-\rho)]^{K-1}\right\} \end{aligned} \tag{25}$$

while the bandwidth constraint (15) becomes

$$\rho \geq \frac{f_K - \gamma^2}{f_K - 1} \tag{26}$$

Since (25) is monotonically decreasing in ρ when $0 \leq \rho \leq 1$, the TC is maximized when ρ achieves equality in (26). Substituting ρ from (26) with equality into (25), and maximizing over all γ, we have (23). ∎

In Figure 1, we plot the lower bound to the $\mathrm{TC}_{\mathrm{EP}}$ for different values of K and Λ. Since the TC under the total power constraint serves as an upper bound to the $\mathrm{TC}_{\mathrm{EP}}$, Figure 1 gives a tight upper and lower bound to the Total Capacity of a equal-power constrained channel, for moderate number of users.

As a performance measure, we define the Total Capacity Ratio (TCR) as the ratio of the K-user TC to K times the single-user capacity with the same RMS bandwidth and average signal-to-noise ratio constraints. Since the single-user capacity of a RMS bandlimited PAM channel is equal to $B\log[1+\Lambda]$ (see [7]), the TCR can be written as

$$\mathrm{TCR}(K,\Lambda) \triangleq \frac{\mathrm{TC}(B,K,\Lambda)}{KB\log[1+\Lambda]} \tag{27}$$

The TCR gives the ratio of the capacity available to an average user (when the channel is shared by K users) to the single-user capacity. In other words, it measures, from the user's viewpoint, the ratio of the average user capacity in a multi-user channel to the capacity in a single-user channel. Notice that the TCR depends only on K and Λ, and is independent of B. Using the lower bound in Theorem 2.2, we obtain a lower bound to the TCR under the equal-power constraint for all signal-to-noise ratios.

Corollary 2.1.

A lower bound to the TCR under the equal-power constraint for all signal-to-noise ratio is

$$\mathrm{TCR}(K,\Lambda) \geq \frac{1}{\gamma} \tag{28}$$

where γ is the positive real root of the equation

$$\gamma(\gamma^2-1) = f_K - 1 \tag{29}$$

Proof.

In order to obtain (28), we simply substitute γ from (29) into (23) and (27). Since there is one and only one real positive solution in (29), there is no ambiguity in the value of γ. ∎

In Figure 2, we show the TCR under the total power constraint and the lower bound to the TCR under the equal-power constraint for different number of users and different average signal-to-noise ratios.

3. EFFICIENCIES

The TCR gives the performance degradation, from the user's viewpoint, when a bandlimited channel is shared by K users instead of a single user. A natural question to be asked is "How to maintain the same rate in the presence of other users?" If we want to maintain the same information rate, we have to modify some of the parameters. In the following, we will analyze two alternatives. First, we increase the signal-to-noise ratio by increasing the power while the bandwidth remains constant. Second, we increase the bandwidth of the channel while the power of each user remains the same.

The power efficiency, denoted by $\eta_P(K,\Lambda)$, is defined as

$$\eta_P(K,\Lambda) \triangleq \frac{e^{\frac{\mathrm{TC}(B,K,\Lambda)}{BK}} - 1}{\Lambda} \tag{30}$$

or, equivalently, implicitly as

$$\mathrm{TC}(B,K,\Lambda) = BK\log[1+\eta_P(K,\Lambda)\Lambda] \tag{31}$$

The bandwidth efficiency, denoted by $\eta_B(K,\Lambda)$, is defined implicitly as

$$\mathrm{TC}(B,K,\Lambda) = \eta_B(K,\Lambda)BK\log[1+\frac{\Lambda}{\eta_B(K,\Lambda)}] \tag{32}$$

The power efficiency, $\eta_P(K,\Lambda)$ (bandwidth efficiency, $\eta_B(K,\Lambda)$) gives the ratio of the effective power (bandwidth) to the actual power (bandwidth) when the actual signal-to-noise ratio is Λ. The actual power (bandwidth) is the power (bandwidth) used in transmission while the effective power (bandwidth) is the corresponding power (bandwidth) needed for a single user channel to achieve the same capacity. In other words, $-10\log[\eta_P(K,\Lambda)]$ gives the power in db that we have to add to each user in order to maintain the single-user capacity. Similarly, $1/\eta_B(K,\Lambda)$ gives the ratio that we have to increase the bandwidth in order to maintain the same information rate.

Theorem 3.1.

The power efficiency satisfies,

$$\lim_{\Lambda\to\infty} \eta_P(K,\Lambda) = 0 \tag{33}$$

A lower bound to the bandwidth efficiency, $\eta_B(K,\Lambda)$, for all signal-to-noise ratio under the equal-power constraint is

$$\eta_B(K,\Lambda) \geq \frac{1}{\sqrt{f_K}} \tag{34}$$

where f_K is defined in (8).

Proof.

From the definition of $\eta_P(K,\Lambda)$, we have,

$$\prod_{n=1}^{N}(1+h_n(\lambda)) = [1+\eta_P(K,\Lambda)\Lambda]^{K\gamma} \tag{35}$$

where N, $h_n(\lambda)$ and γ are all optimally selected for that Λ. Subtracting (14) from (15), it is easy to get

$$h_n(\lambda) \leq \gamma K\Lambda(\gamma^2-1) \qquad n=1,\ldots,N. \tag{36}$$

Substituting (36) into (35), and dividing both side by $\Lambda^{K\gamma}$, we have, in the limit as $\Lambda\to\infty$,

$$\lim_{\Lambda\to\infty} \Lambda^{N-K\gamma}\prod_{n=1}^{N}\left\{\frac{1}{\Lambda}+\gamma K(\gamma^2-1)\right\} \geq \eta_P(K,\Lambda)^{K\gamma} \tag{37}$$

If $\gamma \to 1$ as $\Lambda \to \infty$, the second factor on the left hand side of (37) tends to 0, while the first factor tends either to 0 ($N < K$) or to 1 ($N = K$). On the other hand, if $\gamma \to \alpha > 1$ as $\Lambda \to \infty$, the first factor on the left hand side of (37) tends to 0 for any $N \leq K$, while the second factor is bounded. Therefore, in both cases, the left hand side tends to 0 and since $1 \leq \gamma \leq \sqrt{f_K}$, we have $\eta_P(K, \Lambda) \to 0$ as $\Lambda \to \infty$.

Substituting $\gamma = \sqrt{f_K}$ in Theorem 2.2, we have

$$\mathrm{TC}_{\mathrm{EP}}(B, K, \Lambda) \geq \frac{1}{\sqrt{f_K}} BK \log[1 + \sqrt{f_K}\Lambda] \tag{38}$$

Since the right hand side of (32) is monotonically increasing in $\eta_B(K, \Lambda)$, we have (34) when compared to (38). ∎

The TC is obtained by optimizing the balance between the "symbol rate" factor, B/γ, and the "information sent per symbol" factor, $\log[\cdots]$. As the average signal-to-noise ratio tends to infinity, the "symbol rate" factor dominates and the optimal users' signature waveforms are asymptotically identical. Then, the product term of the signal-to-noise ratios inside the log function in the TC becomes relatively small, and the asymptotic power efficiency is equal to zero. The bandwidth efficiency indicates the increase in bandwidth needed to maintain the same user rate when a single-user channel is shared by K users.

In Figure 3 and 4, we plot the Power and Bandwidth efficiency for different values of K and Λ. Also, in the same graphs, we show the lower bound to the efficiencies for the equal-power constrained channel. It shows that regardless of the signal-to-noise ratios, increasing the bandwidth by a factor of 10, we can accommodate about 50 users on the multi-user channel. This indicates a 80 percent reduction in the bandwidth required by Frequency Division Multiple Access (FDMA). Clearly, the tradeoff is a more complicated demodulating and decoding process in the Synchronous Code Division Multiple Access (CDMA) channel, which is a special case of the current model.

REFERENCES

[1] Verdú, S., "Capacity region of Gaussian CDMA channels: the symbol-synchronous case," *Proc. of the Twenty-fourth Allerton Conference on Communication, Control and Computing*, Allerton, IL, pp. 1025-1034, October, 1986.

[2] Verdú, S., "The capacity region of the symbol-asynchronous Gaussian multiple-access channel," *IEEE Transactions on Information Theory*, vol. IT-35, pp.733-751, July, 1989.

[3] Amoroso, F., "The bandwidth of digital data signals," *IEEE Communication Magazine*, vol. 18, pp. 13-24, November 1980.

[4] Gabor, D., "Theory of communication," *J. IEE*, vol. 93, pt. 3, pp. 429-457, November 1946.

[5] Nuttall, A. H., "Minimum rms bandwidth of M time-limited signals with specified code or correlation matrix," *IEEE Transactions on Information Theory*, vol. IT-14, pp. 699-707, September 1968.

[6] Narayan, P. and D. Snyder, "Signal set design for bandlimited memoryless multiple-access channels with soft decision demodulation," *IEEE Transactions on Information Theory*, vol. IT-33, pp. 539-556, July, 1987.

[7] Cheng, R. S. and S. Verdú, "Optimal Signal Design for Band-Limited PAM Synchronous Multiple-Access Channels," *Proc. of the Twenty-third Annual Conference on Information Sciences and Systems*, Baltimore, ML, pp. 321-326, March, 1989.

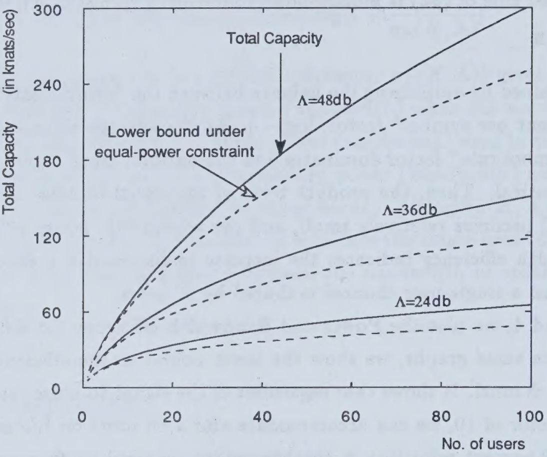


Figure 1. Total Capacity for different number of users and average signal-to-noise ratios when B=1khz

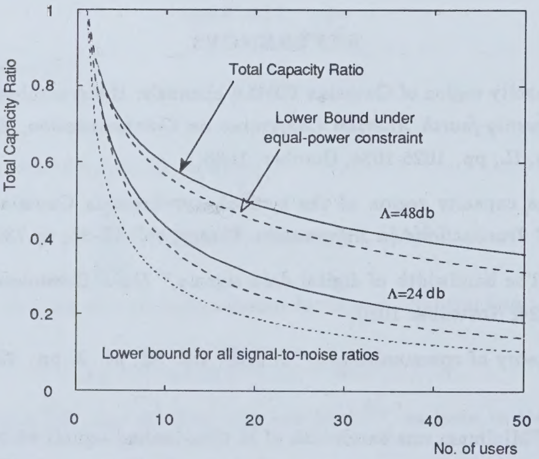


Figure 2. Total Capacity Ratio for different number of users and average signal-to-noise ratios

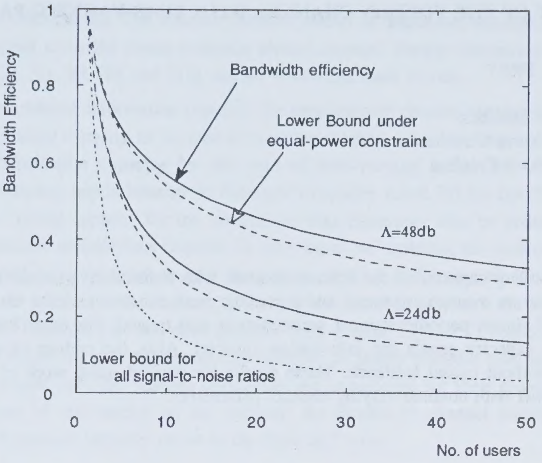


Figure 3. Bandwidth efficiency for different number of users and average signal-to-noise ratios

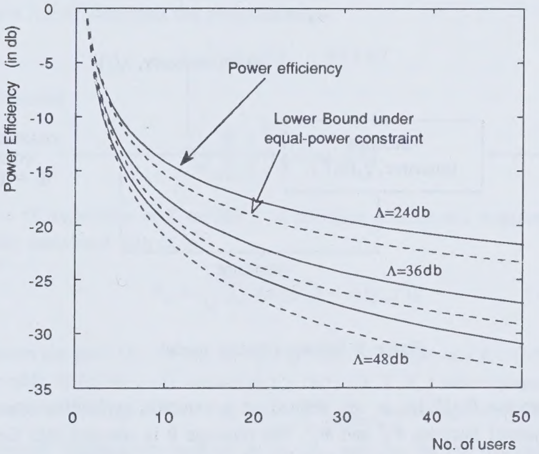


Figure 4. Power efficiency for different number of users and average signal-to-noise ratios

CODING CAPACITY OF THE POISSON CHANNEL WITH TIME-VARYING PARAMETERS

MICHAEL R. FREY

Department of Statistics
University of North Carolina
Chapel Hill, North Carolina

Abstract - The coding capacity of the Poisson channel with nonrandom periodic noise intensity is obtained for an average constraint and a periodic peak constraint on the encoder intensity. The case of almost periodic channel parameters is also treated. For these Poisson channels, the coding capacity equals the information capacity. Also, the coding capacity is the same with and without causal feedback. These results are derived using work of Wyner for the Poisson channel with nontime-varying channel parameters.

I. INTRODUCTION

The Poisson channel model addressed in this paper is represented in Figure 1. The channel output $Y=\{Y_t\}_{0\le t\le T}$ is a Poisson-type (simple) point process [9] directed by the stochastic intensity $\eta_t = \chi_t + \lambda(t)$.

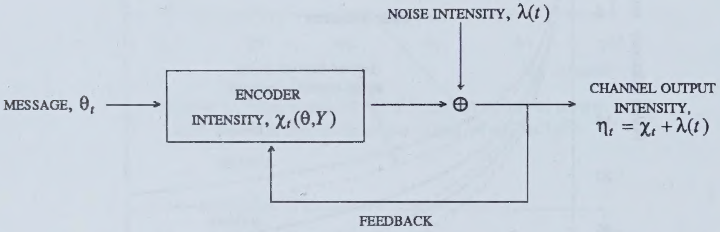

NOISE INTENSITY, $\lambda(t)$
MESSAGE, θ_t
ENCODER
INTENSITY, $\chi_t(\theta,Y)$
$\oplus$
CHANNEL OUTPUT
INTENSITY,
$\eta_t = \chi_t + \lambda(t)$
FEEDBACK

Figure 1. Poisson channel model.

Y and the message process $\theta=\{\theta_t\}_{0\le t\le T}$ are defined on a common probability space $(\Omega,\boldsymbol{F},P)$ with respective completed natural histories $\boldsymbol{F}_t^Y$ and $\boldsymbol{F}_t^\theta$. The message θ is encoded into the channel encoder output via the encoder intensity χ_t. $\chi_t=\chi_t(\theta,Y)$ is required to be an $\boldsymbol{F}_t^\theta \vee \boldsymbol{F}_t^Y$-adapted encoding of the message process θ and channel output Y permitting causal message encoding and noiseless, nonanticipative, instantaneous feedback. The channel noise process N_t is a nonhomogeneous Poisson process with (nonnegative) intensity function $\lambda(t)$. $\lambda(t)$ is sometimes said to represent dark current in the channel model. We take the processes θ and N to be independent and impose, on the encoder intensity, a time-varying peak constraint

$$0 \le \chi_t \le c(t) \tag{1}$$

for all $t \in [0,T]$ where $c(t)$ is positive, bounded, and Lebesgue-measurable and an average constraint

$$E\left[\int_0^T \chi_t\,dt\right] \le k_0 T \tag{2}$$

for some positive constant k_0. The Poisson channel model is elsewhere referred to as the Poisson-type point process channel or as the direct detection photon channel. Further discussion of this channel model can be found in [2], [5], [6], [8], and [12] and the references cited therein.

The Poisson channel information capacity for time-varying channel parameters $\lambda(t)$, $c(t)$ is known, as is the channel coding capacity in the case of nontime-varying noise intensity and nontime-varying peak constraint. The information capacity for the case of time-varying parameters was obtained in [6] by applying simple function approximation to Kabanov's capacity result [8] for the Poisson channel. Wyner [12] obtained the coding capacity for the nontime-varying parameter case by reducing the Poisson channel to a binary discrete memoryless channel. In this paper, we combine the method of approximation by simple functions and Wyner's result to give the coding capacity in the case where the channel parameters are time-varying. In particular, in Theorem 1 we address the case in which the noise intensity $\lambda(t)$ and the peak constraint function $c(t)$ are each periodic with periods T_λ and T_c respectively. Using a similar approach we also obtain the coding capacity for the case of almost periodic channel parameters (Theorem 2). Periodic and almost periodic channel parameters are considered for their physical relevance and also to impose a degree of stationarity on the channel; the classes of channel parameters considered here ensure that the information capacity exists in the limit as $T \to \infty$.

II. CHANNEL CAPACITY DEFINITIONS

Coding capacity is defined within the context of codes, decoding schemes, and decoding error probability. A code (M, T, P_e) for the Poisson channel is a set of M equally likely (nonnegative) waveforms $\chi_m(t)$, $t \in [0,T]$, $m = 1,\ldots,M$ satisfying the peak constraint

$$\chi_m(t) \leq c(t), \qquad 0 \leq t \leq T$$

and the average constraint

$$\frac{1}{M}\sum_{m=1}^{M}\frac{1}{T}\int_0^T \chi_m(t)dt \leq k_0 .$$

Let S_Y be the space of trajectories of Y on $[0,T]$. A decoding scheme is a mapping $D: S_Y \to \{1,2,\ldots,M\}$. The error probability associated with D is

$$P_e = \frac{1}{M}\sum_{1}^{M} P\{D(Y_0^T) \neq m \mid \chi_m(\cdot)\}$$

where $Y_0^T \in S_Y$ denotes the path $\{Y_t : t \in [0,T]\}$. A code (M, T, P_e) has rate $R = (1/T)\ln M$. A code rate R is said to be achievable, if for all $\varepsilon > 0$, there exists a code (M, T, P_e) whose parameters satisfy $M \geq e^{RT}$ with $P_e \leq \varepsilon$ for T sufficiently large. The coding capacity $\mathcal{C}_{\text{CODING}}$ is the supremum of achievable rates.

To define channel information capacity let μ_θ, μ_Y, and $\mu_{\theta Y}$ be the marginal and joint measures induced by the message and output processes, θ and Y, on the path spaces S_θ, S_Y, and $S_\theta \times S_Y$. Write the induced product measure as $\mu_{\theta \times Y}$. Then, the average mutual information in θ and Y over the interval $[0,T]$ is

$$I^T[\theta,Y] = E\left[\ln \frac{d\mu_{\theta Y}}{d\mu_{\theta \times Y}}\right]$$

provided $\mu_{\theta Y} \ll \mu_{\theta \times Y}$; otherwise $I^T[\theta,Y] = \infty$. Expressions for the average mutual information specific to the Poisson channel are given in [9]. The channel information capacity is

$$\mathcal{C}_{\text{INFO}} = \sup_{\theta}\sup_{\chi}\frac{1}{T}I^T[\theta,Y]$$

where θ is any jointly measurable process on $[0,T]$ and $\chi_t = \chi_t(\theta,Y)$ is any $\mathbb{F}_t^{\theta} \wedge \mathbb{F}_t^{Y}$ -adapted mapping.

The information capacity of the Poisson channel with time-varying channel parameters is [6]

$$\mathbb{C}_{\text{INFO}} = \frac{1}{T}\int_0^T C_D(k_0,\lambda(t),c(t))dt\,, \tag{3}$$

where $C_D(k_0,\lambda,c)$ is the information capacity found by Davis [5] for the Poisson channel with nontime-varying channel parameters $\lambda(t)=\lambda$ and $c(t)=c$. We have

$$C_D(x,y,z) = m(x,y,z)\phi(z,y) - \phi(zm(x,y,z),y)\,,$$

$$\phi(x,y) = (x+y)\ln(x+y) - y\ln y\,,$$

and $m(x,y,z) = r_m(x,z) \wedge r_o(y,z)$ is the minimum of $r_m(x,z)$ and $r_o(y,z)$ for $r_m(x,z) = x/z$ and

$$r_o(y,z) = \frac{1}{e}\frac{y}{z}\left[1+\frac{z}{y}\right]^{1+y/z} - \frac{y}{z}\,.$$

$r_o(\lambda(t),c(t))$ can be interpreted as the instantaneous optimum ratio of average to peak signal intensity. $r_m(k_0,c(t))$ is the instantaneous maximum ratio of average to peak signal intensity. In this connection, we note that

$$\frac{1}{e} \le r_o(\lambda(t),c(t)) \le \frac{1}{2}$$

for all $\lambda(t)$, $c(t)$, and k_0.

Fano's inequality provides an upper bound for the coding capacity in terms of the information capacity:

$$\mathbb{C}_{\text{CODING}} \le \limsup_{T\to\infty} \mathbb{C}_{\text{INFO}}\,.$$

It can be shown that, in the case where $\lambda(t)$ and $c(t)$ are periodic with unequal periods, the RHS is just the limit

$$\lim_{T\to\infty} \mathbb{C}_{\text{INFO}} = \frac{1}{T_\lambda T_c}\int_0^{T_\lambda}\int_0^{T_c} C_D(k_0,\lambda(t_\lambda),c(t_c))dt_c\,dt_\lambda\,. \tag{4}$$

Thus, for the case of periodic parameters with $T_\lambda \ne T_c$, the coding capacity is upper bounded by (4).

III. CODING CAPACITY FOR PERIODIC CHANNEL PARAMETERS

Our main result (Theorem 1) is that, for periodic channel parameters $\lambda(t)$ and (bounded) $c(t)$ with respective periods T_λ and T_c, the Poisson channel coding capacity is

$$\mathbb{C}_{\text{CODING}} = \frac{1}{T_\lambda T_c}\int_0^{T_\lambda}\int_0^{T_c} C_D(k_0,\lambda(t_\lambda),c(t_c))dt_c\,dt_\lambda$$

for $T_\lambda \ne T_c$. Thus, in this case, the coding capacity is the same as the channel information capacity [6] as $T \to \infty$.

We find the coding capacity with the aid of several lemmas. First, we construct simple periodic channel parameters $\bar{\lambda}(t)$, $\bar{c}(t)$. Let $A + x = \{t : t - x \in A\}$ and define

$$\bar{E}_j = \bigcup_{k=0}^{\infty}\{E_j + kT_\lambda\}\,, \quad \bar{F}_i = \bigcup_{k=0}^{\infty}\{F_i + kT_c\}$$

where $\{E_j\}$ and $\{F_i\}$ are each Lebesgue-measurable partitions of $[0,T_\lambda]$ and $[0,T_c]$ respectively. Then, we write

$$\tilde{\lambda}(t) = \sum_{j=1}^{n_\lambda} \lambda_j 1_{\bar{E}_j}(t), \quad \tilde{c}(t) = \sum_{i=1}^{n_c} c_i 1_{\bar{F}_i}(t).$$

Second, the channel with simple periodic parameters $\tilde{\lambda}(t)$ and $\tilde{c}(t)$ is transformed by changes of time into an ensemble of $n_\lambda n_c$ parallel Poisson subchannels (Lemma 1). Then, the coding capacity of the original Poisson channel is shown (Lemma 2) to be lower bounded by the sum of the $n_\lambda n_c$ coding capacities $\mathbb{C}^{ij}_{\text{CODING}}$ of the ensemble of parallel Poisson subchannels:

$$\mathbb{C}_{\text{CODING}} \geq \sum_{j=1}^{n_\lambda} \sum_{i=1}^{n_c} \mathbb{C}^{ij}_{\text{CODING}}.$$

The proof of Theorem 1 uses Eq. (4), Wyner's expression [12] for $\mathbb{C}^{ij}_{\text{CODING}}$, and the preceding lemmas to give the result for simple periodic channel parameters. To complete the proof, a limiting argument is used to extend the result to general (not necessarily simple) periodic $c(t)$ and $\lambda(t)$.

Lemma 1: Let

$$\tau_{ij} = \tau_{ij}(t) = \frac{T}{m_{ij}(T)} \int_0^t 1_{\bar{E}_j \cap \bar{F}_i}(s)\,ds \tag{5}$$

where $m_{ij}(T) = |\bar{E}_j \cap \bar{F}_i \cap [0,T]|$. The channel with time $\tau = \tau_{ij}$, message process $\bar{\theta}_\tau = \theta_t$, noise process $\bar{N}_\tau = N_t$, encoder output $\bar{X}_\tau = X_t$, and channel output $\bar{Y}_\tau = \bar{X}_\tau + \bar{N}_\tau$ is a Poisson channel.

Proof: To be able to assert that the channel is Poisson, we need to check that 1) $\bar{Y}_\tau$, $\bar{N}_\tau$, and $\bar{X}_\tau$ are Poisson-type point processes, 2) causality is preserved, and 3) intensities exist for $\bar{\theta}_\tau$ and $\bar{N}_\tau$ and 4) $\bar{\theta}_\tau$ and $\bar{N}_\tau$ are independent processes. But it is well-known [4], [9] that a change of time preserves the Poisson character of a process so requirement 1) is satisfied. $\tau = \tau_{ij}(t)$ is nondecreasing so 2) is satisfied. It is easily checked that $\bar{\theta}_\tau$ and $\bar{N}_\tau$ have intensities - the new intensities are scaled versions of the intensities in the original channel. Also, requirement 3) is obviously satisfied so the assertion of the lemma is valid.

The transformed time $\tau = \tau_{ij}$ is indexed by $i = 1,\ldots,n_c$, $j = 1,\ldots,n_\lambda$. Thus the original channel is transformed into an ensemble of $n_\lambda n_c$ channels. Each of these $n_\lambda n_c$ channels reflects a portion of the events occurring in the original channel and is therefore thought of as a subchannel of the original channel. The ijth subchannel mirrors events occurring in the original Poisson channel during the time $\bar{E}_j \cap \bar{F}_i$. Scaling the time t causes the subchannel intensities to be scaled; the new intensities are $\lambda_{ij} = \lambda_j m_{ij}(T)/T$ and $\chi_t^{ij} = \chi_t m_{ij}(T)/T$. Scaling the encoder intensity in this way is equivalent to scaling the encoder intensity constraints. The constraints in the ijth subchannel are

$$0 \leq \chi_t^{ij} \leq \frac{c_i m_{ij}(T)}{T}, \quad E\left[\int_0^T \chi_t^{ij} dt\right] \leq \frac{k_0 m_{ij}(T)}{T} T.$$

Note that because of the choice of time transformation and the periodicity of $\lambda(t)$ and $c(t)$, the parameters of each subchannel are constant w.r.t. time. We shall want to decompose the Poisson channel into parallel subchannels for the case of $t \in [0,\infty)$. Therefore, the limiting scale factor $m_{ij}(T)/T$ for $T \to \infty$ is needed. In fact,

$$\lim_{T \to \infty} \frac{m_{ij}(T)}{T} = \frac{|E_j|}{T_\lambda} \frac{|F_i|}{T_c}.$$

Lemma 2: Let $\mathbb{C}^{ij}_{\text{CODING}}$ be the coding capacity of the Poisson subchannel formed by the change of time (5). Then

$$\mathbb{C}_{\text{CODING}} \ge \sum_{j=1}^{n_\lambda}\sum_{i=1}^{n_c} \mathbb{C}^{ij}_{\text{CODING}} .$$

Proof: Let R_{ij}, $1 \le i \le n_c$, $1 \le j \le n_\lambda$ be achievable rates for the ensemble of parallel Poisson subchannels. Then there exist codes (M_{ij}, T, P_e^{ij}) such that, for T sufficiently large, $M_{ij} \ge e^{R_{ij}T}$ and $P_e^{ij} \le \varepsilon$. Combination of these codes gives a code (M, T, P_e) on the original channel with

$$M = \prod_{i,j} M_{ij} , \qquad P_e = 1 - \prod_{i,j}(1 - P_e^{ij}) .$$

The rate $R = (1/T)\ln M = \sum_{i,j} R_{ij}$ of this code is achievable since

$$M = \prod_{i,j} M_{ij} \ge \prod_{i,j} e^{R_{ij}T} = e^{RT}$$

and, for all $\varepsilon_1 > 0$, we can make $P_e \le \varepsilon_1$ by choosing T large enough that $P_e^{ij} \le 1 - (1-\varepsilon_1)^{1/n}$. Since the supremum of all achievable rates is no less than the supremum restricted to achievable rates of combined codes, the result is proved.

Theorem 1: Consider the Poisson channel with nonrandom periodic noise intensity $\lambda(t)$ and peak- and average-constrained encoder intensity, as in (1) and (2), with $c(t)$ periodic and bounded. This channel has coding capacity

$$\mathbb{C}_{\text{CODING}} = \frac{1}{T_\lambda T_c} \int_0^{T_\lambda}\int_0^{T_c} C_D(k_0, \lambda(t_\lambda), c(t_c))\, dt_c\, dt_\lambda \tag{6}$$

for $T_\lambda \ne T_c$. For $T_\lambda = T_c$ the coding capacity is

$$\mathbb{C}_{\text{CODING}} = \frac{1}{T_\lambda} \int_0^{T_\lambda} C_D(k_0, \lambda(t), c(t))\, dt .$$

Proof: We only prove the first equality. For $T_\lambda \ne T_c$, it is shown in the Appendix that

$$\lim_{T\to\infty} \frac{m_{ij}(T)}{T} = \frac{|E_j|}{T_\lambda}\frac{|F_i|}{T_c} .$$

So, from Lemma 1 and [12],

$$\begin{aligned}
\mathbb{C}^{ij}_{\text{CODING}} &= C_D(k_0 \lim_{T\to\infty}\frac{m_{ij}(T)}{T}, \lambda_j \lim_{T\to\infty}\frac{m_{ij}(T)}{T}, c_i \lim_{T\to\infty}\frac{m_{ij}(T)}{T}) \\
&= C_D(k_0 \frac{|E_j|}{T_\lambda}\frac{|F_i|}{T_c}, \lambda_j \frac{|E_j|}{T_\lambda}\frac{|F_i|}{T_c}, c_i \frac{|E_j|}{T_\lambda}\frac{|F_i|}{T_c}) \\
&= \frac{|E_j|}{T_\lambda}\frac{|F_i|}{T_c} C_D(k_0, \lambda_j, c_i) .
\end{aligned}$$

Write $\mathbb{C}_{\text{CODING}}(\lambda, c)$ for the capacity of the channel with parameters $\lambda(t)$, $c(t)$. Then, using Lemma 2,

$$\mathbb{C}_{\text{CODING}}(\tilde{\lambda}, \tilde{c}) \ge \sum_{j=1}^{n_\lambda}\sum_{i=1}^{n_c} \frac{|E_j|}{T_\lambda}\frac{|F_i|}{T_c} C_D(k_0, \lambda_j, c_i)$$

$$= \frac{1}{T_\lambda T_c} \int_0^{T_\lambda} \int_0^{T_c} C_D(k_0, \bar{\lambda}(t_\lambda), \bar{c}(t_c)) dt_c \, dt_\lambda .$$

Let $\tilde{\lambda}_n(t)$ belong to a sequence of simple functions converging downward to $\lambda(t)$ and let $\tilde{c}_n(t)$ belong to a similar sequence converging upward to $c(t)$. Since, for each n, $\lambda(t) \leq \tilde{\lambda}_n(t)$ and $c(t) \geq \tilde{c}_n(t)$, we have

$$\mathbb{C}_{\text{CODING}}(\lambda, c) \geq \mathbb{C}_{\text{CODING}}(\tilde{\lambda}_n, \tilde{c}_n), \quad n = 1,2,3,\ldots$$

Then,

$$\mathbb{C}_{\text{CODING}}(\lambda, c) \geq \frac{1}{T_\lambda T_c} \int_0^{T_\lambda} \int_0^{T_c} C_D(k_0, \tilde{\lambda}_n(t_\lambda), \tilde{c}_n(t_c)) dt_c \, dt_\lambda$$

and, by Fatou's Lemma [11],

$$\mathbb{C}_{\text{CODING}}(\lambda, c) \geq \frac{1}{T_\lambda T_c} \int_0^{T_\lambda} \int_0^{T_c} C_D(k_0, \lambda(t_\lambda), c(t_c)) dt_c \, dt_\lambda .$$

This and (4) proves the result.

Wyner [12] showed that the coding capacity of the Poisson channel with constant parameters is not increased by feedback. The foregoing proof preserves this feature of Wyner's result. Hence we find that the coding capacity for the case of periodic channel parameters is the same with or without feedback.

IV. CODING CAPACITY FOR ALMOST PERIODIC CHANNEL PARAMETERS

Definition: A real function $f(t)$ defined on the nonnegative real line is almost periodic if, for any $\varepsilon > 0$, there exists a number $l(\varepsilon) > 0$ with the property that any interval in the nonnegative real line of length $l(\varepsilon) > 0$ contains at least one point ξ such that

$$|f(t+\xi) - f(t)| < \varepsilon, \qquad 0 \leq t < \infty .$$

This is the definition of almost periodic functions given by Bohr [1] (or see [3].) It is easily checked that the function $f(t) = \sin(t) + \sin(\pi t)$ is almost periodic but not periodic. Thus we are motivated to consider almost periodic channel parameters of the Poisson channel.

Consider the preceding definition for $f(t)$ periodic, with period T, say. In any interval of length greater than T, a $\xi = kT$, for some integer k, can be found such that $f(t+\xi) = f(t)$ for all $t \geq 0$. Thus we note that all periodic functions are almost periodic. On need not therefore treat periodic channel parameters as a case separate from that of almost periodic channel parameters. However, in this paper we do consider the two cases separately because of the special form taken by the expression for capacity (Eq. 6) in the case of periodic channel parameters and because this approach makes the case of periodic channel parameters accessible to those readers unacquainted with properties of almost periodic functions.

Theorem 2: Suppose a Poisson channel has almost periodic noise intensity $\lambda(t)$ and average constraint parameter $0 < k_0 \leq \infty$. Suppose the peak constraint function $c(t)$ is almost periodic and bounded. Then the channel coding capacity is the limit

$$\mathbb{C}_{\text{CODING}} = \lim_{T \to \infty} \frac{1}{T} \int_0^T C_D(k_0, \lambda(t), c(t)) dt .$$

Proof: The proof hinges on properties of almost periodic functions. $c(t)$ is assumed to be bounded, therefore

$$\mathbb{C}_{\text{INFO}} = \frac{1}{T}\int_0^T C_D(k_0,\lambda(t),c(t))dt .$$

Both $\lambda(t)$ and $c(t)$ are almost periodic so $C_D(k_0,\lambda(t),c(t))$ is almost periodic. Because $C_D(k_0,\lambda(t),c(t))$ is almost periodic, $\mathbb{C}_{\text{INFO}}$ exists in the limit as $T\to\infty$. Therefore

$$\mathbb{C}_{\text{CODING}} \le \lim_{T\to\infty}\frac{1}{T}\int_0^T C_D(k_0,\lambda(t),c(t))dt . \tag{7}$$

Define the simple functions $\tilde{\lambda}_n(t)$, $\tilde{c}_n(t)$ as follows:

$$\tilde{\lambda}_n(t) = \begin{cases} 2^{-n}\left\lceil 2^n\lambda(t)\right\rceil, & \lambda(t)\le 2^n \\ \infty, & \lambda(t)>2^n \end{cases},$$

$$\tilde{c}_n(t) = \begin{cases} 2^{-n}\left\lfloor 2^n c(t)\right\rfloor, & c(t)\le 2^n \\ 2^n, & c(t)>2^n \end{cases}.$$

We can write

$$\tilde{\lambda}_n(t) = \sum_{j=0}^{2^{2n}} 1_{E_j}(t)\lambda_j , \quad \tilde{c}_n(t) = \sum_{i=0}^{2^{2n}} 1_{F_i}(t)c_i$$

where

$$\lambda_0 = \infty, \quad E_0 = \{t:\lambda(t)>2^n\}, \quad c_0 = 2^n, \quad F_0 = \{t:c(t)>2^n\},$$

and for $j>0$, $i>0$,

$$\lambda_j = \frac{j}{2^n}, \quad E_j = \{t:\tilde{\lambda}_n(t)=\lambda_j\},$$

$$c_i = \frac{i-1}{2^n}, \quad F_i = \{t:\tilde{c}_n(t)=c_i\}.$$

Define $m_{ij}(T) = |[0,T]\cap E_j\cap F_i|$. $\lambda(t)$ and $c(t)$ are almost periodic so the limits

$$\alpha_{ij} = \lim_{T\to\infty}\frac{m_{ij}(T)}{T}$$

exist. For i, j such that $\alpha_{ij}\ne 0$, define the changes of channel time

$$\tau_{ij}(t) = \frac{1}{\alpha_{ij}}\int_0^t 1_{E_j\cap F_i}(s)ds .$$

These changes of channel time transform the channel with parameters $\tilde{\lambda}_n(t)$, $\tilde{c}_n(t)$ into (up to) 2^{4n} Poisson subchannels - each with constant channel parameters. $\alpha_{ij}=0$ corresponds to a subchannel with a peak constraint function which is identically zero; such subchannels have zero coding capacity and can be ignored. Also certain subchannels may have infinite noise intensity; these subchannels also have zero coding capacity. $C_D(\cdot,\cdot,0)=C_D(\cdot,\infty,\cdot)=0$ so we have

$$\mathbb{C}_{\text{CODING}}(\tilde{\lambda}_n,\tilde{c}_n) \ge \sum_{i=0}^{2^{2n}}\sum_{j=0}^{2^{2n}} C_D(\alpha_{ij}k_0,\alpha_{ij}\lambda_j,\alpha_{ij}c_i)$$

$$= \sum_{i=0}^{2^{2n}} \sum_{j=0}^{2^{2n}} \alpha_{ij} C_D(k_0, \lambda_j, c_i)$$

$$= \lim_{T \to \infty} \frac{1}{T} \sum_{i=0}^{2^{2n}} \sum_{j=0}^{2^{2n}} C_D(k_0, \lambda_j, c_i) m_{ij}(T)$$

$$= \lim_{T \to \infty} \frac{1}{T} \int_0^T C_D(k_0, \tilde{\lambda}_n(t), \tilde{c}_n(t)) dt \, .$$

$\tilde{\lambda}_n(t)$, $\tilde{c}_n(t)$ were defined so that $\lambda(t) \le \tilde{\lambda}_n(t)$ and $c(t) \ge \tilde{c}_n(t)$. Therefore (Appendix 3),

$$\mathbb{C}_{\text{CODING}}(\lambda, c) \ge \mathbb{C}_{\text{CODING}}(\tilde{\lambda}_n, \tilde{c}_n)$$

for all n. Then, using Moore's theorem [7] (Appendix 1),

$$\mathbb{C}_{\text{CODING}}(\lambda, c) \ge \lim_{n \to \infty} \mathbb{C}_{\text{CODING}}(\tilde{\lambda}_n, \tilde{c}_n)$$

$$\ge \lim_{n \to \infty} \lim_{T \to \infty} \frac{1}{T} \int_0^T C_D(k_0, \tilde{\lambda}_n(t), \tilde{c}_n(t)) dt$$

$$= \lim_{T \to \infty} \frac{1}{T} \lim_{n \to \infty} \int_0^T C_D(k_0, \tilde{\lambda}_n(t), \tilde{c}_n(t)) dt$$

$$= \lim_{T \to \infty} \frac{1}{T} \int_0^T C_D(k_0, \lambda(t), c(t)) dt \, .$$

This and (7) combine to complete the proof.

Observations made in the last section with respect to coding capacity for periodic channel parameters apply as well to the almost periodic case: the coding capacity is the same with and without feedback and the coding capacity is the same as the information capacity in the limit as $T \to \infty$,

$$\mathbb{C}_{\text{CODING}} = \lim_{T \to \infty} \mathbb{C}_{\text{INFO}} \, .$$

The reader has perhaps noted the similarity of the approaches used to obtain the coding capacity in the case of periodic channel parameters and in the case of almost periodic channel parameters. From a close inspection of the two approaches, it is clear that expressions for the coding capacity can be given in cases of mixed channel parameter types: periodic $c(t)$ and almost periodic $\lambda(t)$ and vice versa. For instance, for $c(t)$ periodic (and bounded) and $\lambda(t)$ almost periodic

$$\mathbb{C}_{\text{CODING}} = \lim_{T_\lambda \to \infty} \frac{1}{T_c T_\lambda} \int_0^{T_c} \int_0^{T_\lambda} C_D(k_0, \lambda(t_\lambda), c(t_c)) dt_c \, dt_\lambda \, .$$

The class of Bohr-almost periodic functions treated here is not the largest known class of almost periodic functions. In particular, the Bohr-almost periodic functions do not include all Stepanoff-almost periodic functions [10], [11] (or see [3]). For channel parameters of this latter class, we conjecture that

$$\mathbb{C}_{\text{CODING}} = \lim_{T \to \infty} \frac{1}{T} \int_0^T C_D(k_0, \lambda(t), c(t)) dt$$

and, also,

$$\mathbb{C}_{\text{CODING}} = \lim_{T_c \to \infty} \lim_{T_\lambda \to \infty} \frac{1}{T_c T_\lambda} \int_0^{T_c} \int_0^{T_\lambda} C_D(k_0, \lambda(t_\lambda), c(t_c)) dt_c \, dt_\lambda \, .$$

Proposition: Let $m(T) = |\bar{E} \cap \bar{F} \cap [0,T]|$ where

$$\bar{E} = \bigcup_{k=0}^{\infty} \{E + kT_\lambda\}, \quad \bar{F} = \bigcup_{k=0}^{\infty} \{F + kT_c\}$$

and where E and F are Lebesgue-measurable subsets of $[0,T_\lambda]$ and $[0,T_c]$ respectively with $T_\lambda \neq T_c$. Then

$$\lim_{T\to\infty} \frac{m(T)}{T} = \frac{|E|}{T_\lambda} \frac{|F|}{T_c}.$$

Proof: Without loss of generality, take $T_\lambda < T_c$. Define $\bar{E}(\rho) = \bar{E} - \rho$ for each $\rho \in [0,T_\lambda]$. Let Q be the set of $\rho \in [0,T_\lambda]$ for which there exists a T_ρ such that, for all $t \geq 0$,

$$1_{\bar{E}(\rho)\cap\bar{F}}(t + T_\rho) = 1_{\bar{E}\cap\bar{F}}(t).$$

Q is dense in $[0,T_\lambda]$. Thus, for each $\rho \in [0,T_\lambda]$, there exists a T'_ρ such that

$$1_{\bar{E}(\rho)\cap\bar{F}}(t + T'_\rho) = 1_{\bar{E}\cap\bar{F}}(t)$$

for all $t \in X_T \subset [0,T]$ where $|[0,T] - X_T| < 1$. This follows from the fact (Littlewood's first principle) that the set $\bar{E}(\rho) \cap \bar{F} \cap [0,T]$ is very nearly a finite union of open intervals. For each $\rho \in [0,T]$, we have

$$\lim_{T\to\infty} \frac{|\bar{E}(\rho)\cap\bar{F}\cap[0,T]|}{T} = \lim_{T\to\infty} \left[\frac{1}{T} \int_0^{T'_\rho} 1_{\bar{E}(\rho)\cap\bar{F}}(t)dt + \frac{T - T'_\rho}{T} \frac{1}{T - T'_\rho} \int_{T'_\rho}^{T} 1_{\bar{E}(\rho)\cap\bar{F}}(t)dt \right]$$

$$= \lim_{T\to\infty} \frac{1}{T - T'_\rho} \int_{T'_\rho}^{T} 1_{\bar{E}(\rho)\cap\bar{F}}(t)dt$$

$$= \lim_{T\to\infty} \frac{1}{T - T'_\rho} \int_0^{T-T'_\rho} 1_{\bar{E}(\rho)\cap\bar{F}}(s+T'_\rho)ds$$

$$= \lim_{T\to\infty} \frac{1}{T} \int_0^{T} 1_{\bar{E}(\rho)\cap\bar{F}}(s+T'_\rho)ds$$

$$= \lim_{T\to\infty} \frac{1}{T} \left\{ \int_0^{T} 1_{\bar{E}\cap\bar{F}}(s)ds + \int_{[0,T]-X_T} \left[1_{\bar{E}(\rho)\cap\bar{F}}(s+T'_\rho) - 1_{\bar{E}\cap\bar{F}}(s) \right] ds \right\}.$$

Now

$$\left| \int_{[0,T]-X_T} \left[1_{\bar{E}(\rho)\cap\bar{F}}(s+T'_\rho) - 1_{\bar{E}\cap\bar{F}}(s) \right] ds \right| \leq 2|[0,T] - X_T| \leq 2$$

so

$$\lim_{T\to\infty} \frac{|\bar{E}(\rho)\cap\bar{F}\cap[0,T]|}{T} = \lim_{T\to\infty} \frac{1}{T} \int_0^{T} 1_{\bar{E}\cap\bar{F}}(t)dt = \lim_{T\to\infty} \frac{m(T)}{T}.$$

Let R be a random variable, uniformly distributed over the interval $[0,T_\lambda]$. By the Bounded Convergence theorem,

$$\lim_{T\to\infty} \frac{m(T)}{T} = E\left[\lim_{T\to\infty} \frac{|\bar{E}(R)\cap\bar{F}\cap[0,T]|}{T} \right]$$

$$= \lim_{T\to\infty} \frac{1}{T} \int_0^T \int_0^{T_\lambda} 1_{\bar{E}(R)\cap\bar{F}}(t)\,dR\,dt$$

$$= \lim_{T\to\infty} \frac{1}{T} \int_0^T \int_0^{T_\lambda} 1_{\bar{E}(R)}(t)\,dR\ 1_{\bar{F}}(t)\,dt$$

$$= \lim_{T\to\infty} \frac{1}{T} \int_0^T \frac{|E|}{T_\lambda} 1_{\bar{F}}(t)\,dt$$

$$= \frac{|E|}{T_\lambda} \frac{|F|}{T_c} .$$

REFERENCES

[1] H. Bohr, "Sur les fonctions presque-périodiques," *C. R. Acad. Sci. Paris*, Vol. 177, p. 737-739, 1923.

[2] P. Brémaud, *Point Processes and Queues*, Springer-Verlag, New York, 1981.

[3] C. Corduneanu, *Almost Periodic Functions*, Interscience Publishers, New York, 1968.

[4] D.J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*, Springer-Verlag, New York, 1988.

[5] M. Davis, "Capacity and cut-off rate for Poisson-type channels," *IEEE Transactions on Information Theory*, Vol. 26, No. 6, pp. 710-715, 1980. No. 1, 1974.

[6] M.R. Frey, "Capacity of the Poisson Channel with Time-Varying noise Intensity and Jamming," LISS Technical Report, No. 25, University of North Carolina, May 1988.

[7] L.M. Graves, *The Theory of Functions of Real Variables*, McGraw-Hill, New York, 1946.

[8] Y. Kabanov, "The capacity of a channel of the Poisson type," *Theory of Probability and Applications*, Vol. 23, 1974.

[9] R. Liptser and A. Shiryayev, *Statistics of Random Processes, Vol. II, Applications*, Springer-Verlag, New York, 1977.

[10] W.W. Stepanoff, "Sur quelques généralisations des fonctions presque-périodiques," *C. R. Acad. Sci. Paris*, Vol. 181, pp. 90-94, 1925.

[11] W.W. Stepanoff and A. Tychonoff, "Sur les espaces des fonctions presque-périodiques," *C. R. Acad. Sci. Paris*, Vol. 196, pp. 1199-1201, 1933.

[12] A.D. Wyner, "Capacity and Error Exponent for the Direct Detection Photon Channel - Part I, II," *IEEE Trans. on Info. Theory*, Vol. 34, No. 6, November 1988.

Multiple Source Signals Separation in Noise

L. Tong, V. C. Soon, Y. F. Huang, and R. Liu
Department of Electrical & Computer Engineering, University of Notre Dame
Notre Dame, IN. 46556

Abstract
The problem of multiple source signals separation in noise is considered in this paper. An eigenstructure based algorithm called Extended Fourth Order Blind Identification (EFOBI) is proposed and shown by simulation to have better performance than the previously proposed algorithm called Fourth Order Blind Identification(FOBI) [Cardoso89], which fails to account for noise. EFOBI is applied to the high resolution bearing estimation problem with a linear array of sensors with uncertain sensor gains. Simulation shows that EFOBI is robust with respect to such uncertainties.

I. INTRODUCTION

Separating source signals from multi-channel measurements is a problem encountered frequently in various signal processing applications. Observations collected from multiple sensors are usually composed of signals from several independent sources traveling through some unknown medium or channel. Moreover, the received signals are generally corrupted by noise of different kinds such as measurement noise and environmental noise. In many cases, as shown in the schematic diagram of Figure 1, only the received signals at the sensors are available while the source signals and the characteristics of the medium are unknown, or at least partially unknown.

Many practical signal processing applications belong to this category. Some of those are listed in the following:

(i) *Decomposition of Multi-neuron Signals* [Abeles77], [Gerstein83], [Salgonicoff88], [Schmidt84]. For many years, neuro-biologists have been trying to unravel secrets of behavior of the individual neuron by studying the neuronal firing signals collected from the brain. To simplify analysis, it has been commonly assumed that the wave shape of the neuronal signals are known, and there is no overlapping between different signals generated from different neurons. Practically, however, measurements usually consist of overlapped signals of unknown shape from a group of neurons rather than a single neuron. This overlapping makes it extremely difficult to analyze the behavior of individual neurons. The major problem of multi-neuron analysis is to separate neural signals from each other. Some heuristic approach to separate overlapping signals has been proposed [Abeles77] with limited success.

(ii) *Speech Enhancement* . In a multi-speaker environment, the speech signals may consist of speech signals from several speakers or other unknown sources. For speech to be recognized, it is desirable that the individual speech signals are separated. Another practical problem is that the speech signals of pilots of an airplane, or speech signals collected from noisy production line are heavily corrupted by low frequency engine noise. This makes the communication between pilots and ground stations or between two pilots difficult. Since the engine noise varies among different planes, and voice signals varies among pilots, it is unlikely to design a filter that provides satisfactory results for all situations.

(iii)*Radar Array and Sonar Array Signal Processing*, such as bearing estimation and spectral estimation [Kay88]. Here, the source signals are usually assumed to be complex sinusoidal waves with unknown amplitudes, frequencies and phases. Many algorithms such as Pisarenko Harmonic Decomposition (PHD) [Pisarenko73], and MUSIC [Schmidt81] have been proposed and studied. These algorithms usually do not perform well when near-field phase effects are present [Cardoso89], or when the array of sensors has uncertainties in the sensor gain and phases. This uncertainty in sensor gains and phases will cause the existing eigenstructure methods like MUSIC to have a degraded performance.

Recently, some innovative approaches to multiple source separation problem in signal processing have been proposed by J. Herault and C. Jutten [Herault86] and then by Cardoso [Cardoso89]. By employing stochastic gradient descent algorithm, Herault and Jutten proposed an adaptive procedure which has a structure suitable for VLSI implementation [Vittoz89]. Their approach, however, has several disadvantages. First of all, the algorithm is derived from a two-signal-source and two-sensor case. The extension to general multiple sources, multiple sensors is nontrivial. The performance of the algorithm in the general case is difficult to assess. Secondly, the algorithm does not guarantee convergence even in the two source signals and two sensors case. Finally, the algorithm deals only with noise-free case. Cardoso later proposed an elegant solution, the so called Fourth Order Blind Identification (FOBI), to the multiple source separation problem in a noise-free environment. Fourth order statistics and eigenstructure methods are used in Cardoso's FOBI algorithm. Simulation shows that FOBI outperforms the acclaimed MUSIC algorithm in bearing estimation when near-field phase distortion effects are present in the received signals [Cardoso89]. Both of the above methods assume that the received signals are noiseless which is rarely the case in most signal processing applications.

In this paper, Cardoso's FOBI algorithm is extended to encounter noisy signals. The resulting algorithm is referred to as Extended FOBI (EFOBI) here and is presented in Section II. EFOBI is compared with FOBI at different noise level through a simulation example. The EFOBI is then applied to the problem of high resolution bearing estimation under sensor gain uncertainties with a linear array of sensors.

II EXTENDED FOBI ALGORITHM

In this section, we first present the mathematical formulation of the Multiple Source Separation (MSS) problem and state the assumptions on which EFOBI is based. An outline of the algorithm and its derivation are then given.

II.A. *Mathematical Model and Assumptions*

The mathematical model of the MSS problem can be formulated as follows:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) + \mathbf{n}(t) \qquad t = 1,2,... \qquad (1)$$

where $\mathbf{x}(t) \in R^{n\times 1}$ is the measurement vector from n sensors, $\mathbf{s}(t) \in R^{m\times 1}$ is a vector of m unknown signal sources, $\mathbf{A} \in R^{n\times m}$ characterizes the medium or channel, $\mathbf{n}(t) \in R^{n\times 1}$ is the additive random noise vector. The objective of MSS is to estimate $\mathbf{s}(t)$ and $\mathbf{A}$ from samples of $\mathbf{y}(t)$. Obviously, the solutions of $\mathbf{s}(t)$ and $\mathbf{A}$ are not unique since $\mathbf{s}(t)$ and $\mathbf{A}$ can be scaled arbitrarily without affecting $\mathbf{y}(t)$. A more sensible objective is thus to estimate $\mathbf{s}(t)$ and columns of $\mathbf{A}$ up to a scale factor. Throughout this paper, the following assumptions are made:

(A1) Each element of $\mathbf{s}(t)$ is a zero mean, stationary process. Without loss of generality (by scaling columns of A), it is assumed that the i-th component of s satisfies

$$E(s_i^2(t)) = 1, \text{ for } i = 1, 2, ..., m.$$

(A2) Each component of s has different fourth order statistical moment, i.e.,

$$E(s_i^4(t)) \neq E(s_j^4(t)), \text{ for } i \neq j.$$

(A3) $\mathbf{n}(t)$ is a zero mean, identically distributed Gaussian noise with covariance matrix $\sigma^2\mathbf{I}$ where σ^2 is unknown.

(A4) Signals $s_i(t)$ are mutually independent and are also independent of the noise $n_i(t)$ at each sensor, for all t.

(A5) The number of sources m can be either estimated from the singular values [Konstantinos88], [Wax85] and [Kumaresan83], or assumed to be known. Also, it is assumed that n > m.

(A6) $\mathbf{A}$ has a full column rank, i.e., rank($\mathbf{A}$) = m.

II.B. *Extended Forth Order Blind Identification Algorithm (EFOBI)*

The EFOBI algorithm is outlined as follows:

(1) Estimate the covariance matrix $\mathbf{R_x} = E(\mathbf{x}\mathbf{x}^t)$

(2) Compute the Singular Value Decomposition (SVD) of $\mathbf{R}_x$. It can be shown that the SVD of $\mathbf{R}_x$ has the following form:

$$\mathbf{R}_x = \mathbf{U}_s diag(\lambda_1^2,...,\lambda_m^2)\mathbf{U}_s^t + \sigma^2\mathbf{I} \qquad (2)$$

$$= [\mathbf{U}_s \mathbf{U}_n] diag(\lambda_1^2 + \sigma^2,...,\lambda_m^2 + \sigma^2, \sigma^2,...,\sigma^2)\begin{bmatrix}\mathbf{U}_s^t \\ \mathbf{U}_n^t\end{bmatrix} \qquad (2')$$

Where $\mathbf{U}_s, \mathbf{U}_n$ are the eigenvectors corresponding to signals (plus noise) subspace and noise (only) subspace respectively.

(3) Data transformation:

Let $\mathbf{C} =: diag(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_m})\mathbf{U}_s^t$, $\mathbf{B} = \mathbf{CA}$ and $\mathbf{w} = \mathbf{Cn}$

$$\mathbf{y} =: \mathbf{Cx} = \mathbf{Bs} + \mathbf{w} \tag{3}$$

$$\mathbf{z} =: \|\mathbf{y}\|\mathbf{y} \tag{4}$$

(4) Estimate the covariance matrix $\mathbf{R}_z = E(\mathbf{z}^t\mathbf{z})$.

(5) Compute $\tilde{\mathbf{R}}_z =: \mathbf{R}_z - \Delta\mathbf{R}_z$, where

$$\Delta\mathbf{R}_z =: diag\{\sum_{l=1}^{m}\frac{\sigma^2}{\lambda_l^2} + \frac{(m+4)\sigma^2}{\lambda_1^2} + \frac{\sigma^4}{\lambda_1^2}(\sum_{l=1}^{m}\frac{1}{\lambda_l^2} + \frac{2}{\lambda_1^2}), \ldots$$
$$\ldots, \sum_{l=1}^{m}\frac{\sigma^2}{\lambda_l^2} + \frac{(m+4)\sigma^2}{\lambda_m^2} + \frac{\sigma^4}{\lambda_m^2}(\sum_{l=1}^{m}\frac{1}{\lambda_l^2} + \frac{2}{\lambda_m^2})\}. \tag{5}$$

(6) Compute the eigenvectors of $\tilde{\mathbf{R}}_z$ denoted as $\mathbf{v}_1, \mathbf{v}_2, \ldots \mathbf{v}_m$

let $\mathbf{V} =: [\mathbf{v}_1\mathbf{v}_2 \ldots \mathbf{v}_m]$ (6)

(7) Estimate the source signals s:

$$\hat{\mathbf{s}} = \mathbf{V}^t\mathbf{y} \tag{7}$$

(8) Estimate A:

$$\hat{\mathbf{A}} = \mathbf{U}_s diag(\lambda_1, \ldots, \lambda_m)\mathbf{V} \tag{8}$$

Remark: In practice, sample covariance matrix will be used in place of $\mathbf{R}_z$ and $\mathbf{R}_x$. In general, the last n - m singular values will not be the same [Okamoto73]. However, σ^2 can be estimated by using algorithms such as [Wax85].

C. *Algorithm Derivation*

The implementation of the algorithm can be summarized as follows: **A** is solved by first transforming **A** into an orthogonal matrix **B**. The transformation is based on the covariance matrix of the observation. Then a matrix comprised of fourth order statistics is constructed. The eigenvectors of this matrix are the columns of **B**. **A** is then solved from **B**. We now prove the following:

Theorem 1. Let **B**, **V and** $\tilde{\mathbf{R}}_z$ be defined as in (3), (6) and (5), respectively. With (A1) - (A6), each column of **B** is an eigenvector of $\tilde{\mathbf{R}}_z$. In other words, $\mathbf{B} = \mathbf{VP}$ where $\mathbf{P}$ is a permutation matrix.

Proof: From (2), we have $\mathbf{AA}^t = \mathbf{U}_s diag(\lambda_1^2, \ldots, \lambda_m^2)\mathbf{U}_s^t$. It is straightforward from the definition of **B** that **B** is an orthogonal matrix. Now

$$\|\mathbf{y}\|^2 = \mathbf{s}^t\mathbf{s} + 2\mathbf{s}^t\mathbf{w} + \mathbf{w}^t\mathbf{w}$$

and

$$\mathbf{y}\mathbf{y}^t = \mathbf{B}\mathbf{s}\mathbf{s}^t\mathbf{B}^t + \mathbf{s}\mathbf{w}^t + \mathbf{w}\mathbf{s}^t + \mathbf{w}\mathbf{w}^t$$

Then $\mathbf{R_z} = E(\mathbf{zz}^t)$

$$= E(\| \mathbf{y} \|^2 \mathbf{yy}^t)$$

with (A1), (A3), A(4) and straightforward manipulations

$$\mathbf{R_z} = \mathbf{B}E(\mathbf{ss}^t\, \mathbf{ss}^t)\mathbf{B}^t + E(\mathbf{s}^t\mathbf{sww}^t) + 2E(\mathbf{ss}^t\, \mathbf{ww}^t) + 2E(\mathbf{ww}^t\mathbf{ss}^t)$$

$$= \mathbf{B}\mathrm{diag}(E(s_1^4)+m-1, \ldots, E(s_m^4)+m-1)\mathbf{B}^t + \Delta\mathbf{R_z}$$

Hence each column of **B** is a eigenvector of $\mathbf{R_z}-\Delta\mathbf{R_z}$ or $\mathbf{B} = \mathbf{VP}$.

The estimate of **A** is obtained from **B** through Eq. (3). Note that **C** is a m $\underline{\underline{\times}}$ n matrix with n > m. **A** is not uniquely solvable in general. However, we shall show that the estimate produced by EFOBI is the "best". The key is that the estimates not only has to satisfy Eq (3), but also has to satisfy certain constraints on the covariance matrix $\mathbf{R_x}$.

Theorem 2. $\hat{\mathbf{A}}$ in Step 8 of EFOBI is the unique estimate of **A** that satisfies the following:

$$\mathbf{V} = \mathbf{C}\,\hat{\mathbf{A}} \qquad (9)$$

$$\hat{\mathbf{A}}\,\hat{\mathbf{A}}^t = \mathbf{R_x} - \sigma^2\mathbf{I} = \mathbf{U}_s \mathit{diag}(\lambda_1^2, \ldots, \lambda_m^2)\mathbf{U}_s^t. \qquad (10)$$

where $\mathbf{C} =: \mathit{diag}(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_m})\mathbf{U}_s^t$

Proof: $\hat{\mathbf{A}}$ in (8) obviously satisfies (9) and (10). We now show it is the unique solution. Assume **T** is any estimate of **A** satisfy (9) and (10), i.e.

$$\mathbf{V} = \mathbf{C}\,\mathbf{T}$$

$$\mathbf{TT}^t = \mathbf{R_x} - \sigma^2\mathbf{I} = \mathbf{U}_s \mathit{diag}(\lambda_1^2, \ldots, \lambda_m^2)\mathbf{U}_s^t.$$

By multiplying **C** on both side of (10), we have

$$\mathbf{VT}^t = \mathit{diag}(\lambda_1, \ldots \lambda_m)\, U_s^{+}$$

Since **V** is orthogonal,

$$\mathbf{T}^t = \mathbf{V}^t\, \mathit{diag}(\lambda_1, \ldots \lambda_m)\mathbf{U}_s^t = \mathbf{A}^t$$

Remarks: (A5) assumes that all the eigenvalues of $\tilde{\mathbf{R}}_z$ are different to insure the unique correspondence between **B** and **V**.. In the case that the eigenvalues of $\tilde{\mathbf{R}}_z$ have multiplicity greater than one, the relation between **B** and **V** is no longer uniquely defined by $\mathbf{B} = \mathbf{PV}$. In this case, those columns of **V** that correspond to the eigenvalues of mutiplicity 1 can still be identified with some columns of **B**.

III PERFORMANCE OF EFOBI

In this section, we evaluate the performance of EFOBI, Comparing EFOBI to FOBI. A Monte Carlo simulation is conducted for various signal to noise ratios (SNR) with 100 trials for each SNR. In the simulation, **A** is chosen as:

$$\mathbf{A} = \begin{bmatrix} 1.0 & 0.3 \\ 0.2 & 1.0 \\ 0.4 & 0.6 \end{bmatrix}$$

$s = [s_1\ s_2]^t$, where s_1 is a zero-mean, unity variance Gaussian signal source and s_2 is a zero-mean, unit variance, uniformly distributed signal source. The noise vector **n**(t), is zero-mean white Gaussian with its variance varied according to different SNR's. 5000 time samples of $\mathbf{x}(t)$ are used for each trial. **A** is then estimated by using FOBI and EFOBI respectively. Since each column of **A** is estimated up to a scale factor, **A** and its estimate are normalized so that the estimation error can be defined. The mean square error (MSE) of the estimate is defined as

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \| \mathbf{A} - \hat{\mathbf{A}}_{(i)} \|_F$$

where N is the number of Monte Carlo trials (100 in our example) and $\| \, . \, \|_F$ denotes the Frobenious norm. Because the estimate $\hat{\mathbf{A}}$ is actually the permuted version of **A**, the columns of are rearranged as the following for comparison purpose: the column in $\hat{\mathbf{A}}$ that is the closest in terms of angle to the i-th column is put in the i-th column of the estimate. Since the variances of the two signals are both equal to one, we define the SNR as:

$$\text{SNR} = 10 \log_{10} 1/\sigma^2 \ \text{(dB)}$$

where is the noise variance. Fig. 2 is a plot of 10 $\log_{10}$ 1/ MSE of EFOBI and FOBI versus SNR. EFOBI performs consistently better than FOBI for various SNRs.

IV APPLICATION: Bearing Estimation with Uncertain Sensor Gain

We examine the problem of high resolution bearing estimation of far field targets in array signal processing when there exist uncertainties in the sensor gains. These uncertainties arise due to the fact that the sensors are not identical or are not calibrated. Eigenstructure methods such as MUSIC, which is one of the most popular algorithms for high resolution bearing estimates, display markedly degraded performance when such uncertainties are present, giving biased estimation and generally poor resolution. Moreover, the MUSIC algorithm requires a searching procedure which affects both the resolution of the estimates and the speed of implementation. Eigenstructure methods for bearing estimation under sensor uncertainties have been studied by Paulraj and Kailath [Paulraj85] and by Friedlander and Weiss [Friedlander88]. Like MUSIC algorithm, these algorithms entail searching through the parameter space. In addition, the algorithm proposed by Paulraj and Kailath provides bearing estimates only up to an unknown arbitrary rotation factor. As we shall see, EFOBI can be applied to this problem and offers the advantage of avoiding the problem of searching through the parameter space.

Bearing estimation problems can be formulated as follows: For each snap shot

$$x_k = \sum_{i=1}^{m} \gamma_k \exp(-j(k-1)\frac{2\pi d}{\lambda}\sin\theta_i)s_i + n_k \qquad k = 1, 2, \ldots, n$$

where x_k is the output of the kth sensor in a n-sensor array, m denotes the number of sources or targets and γ_k denotes the unknown senor gain at the kth sensor, s_i is the i-th source signal, n_k is additive Gaussian noise with unknown variance σ^2, d and λ represent the sensor spacing and the wavelength of the narrow-band source signals respectively. The problem is to estimate the bearing angles $\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_m]'$. Rewriting the equation above in matrix form, we have

$$\mathbf{x}(l) = \boldsymbol{\Gamma}\mathbf{A}(\boldsymbol{\theta})\mathbf{s}(l) + \mathbf{n}(l), \quad l = 1, 2, ..., L.$$

where L denotes the number of snapshots, and $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma_1, \gamma_2, ..., \gamma_m)$, $\mathbf{A}(\boldsymbol{\theta})$ is the steering matrix. With reference to Eq (1), we can estimate $\boldsymbol{\Gamma}\mathbf{A}(\boldsymbol{\theta})$ by applying EFOBI. Note that $\boldsymbol{\Gamma}$ is a real diagonal matrix and every element of the steering matrix $\mathbf{A}(\boldsymbol{\theta})$ has unity magnitude. $\boldsymbol{\theta}$ can be solved from the angle of $\boldsymbol{\Gamma}\mathbf{A}(\boldsymbol{\theta})$.

In the simulation, an equally spaced linear array of eight sensors is used with two sources in far field, i.e. $n = 8$, $m = 2$. The spacing parameter d is set to $\lambda/2$. The two independent targets are placed at bearing angles 25 degrees and 30 degrees, both with unity power. The sensor gains are set to have a nominal value of 1.0 but with uncertainties that vary the actual gains by up to 100% randomly. 300 snapshots are then taken and EFOBI and MUSIC are applied using these snapshots. Fig. 3 shows a histogram of results of Monte Carlo simulation with 1000 trials at a SNR of 30dB. Table 1 shows the means and mean square errors of EFOBI and MUSIC algorithm at SNR values of 10dB,30dB and 60dB. From Figure 3, we can see that the MUSIC bearing estimates are biased, while EFOBI provides estimates with mean values close to the true angles of 25 and 30 degrees. Although EFOBI bearing estimates appears to have higher variances than MUSIC, EFOBI can still provide better performance, since it can be seen from Fig. 3 that even in the worst case, EFOBI estimates are closer to the true bearing angles than MUSIC estimates. This observation is also apparent from an examination of the root mean square error (RMSE) of the estimates in Table 1.b. EFOBI bearing angle estimates have a lower RMSE than MUSIC estimates.

V CONCLUSIONS

The problem of multiple source signals separation is formulated. An eigenstructure based algorithm called Extended Fourth Order Blind Identification (EFOBI) is proposed as a method to solve the problem and shows better performance than another algorithm called Fourth Order Blind Identification(FOBI) [Cardoso89]. EFOBI is then applied to the high resolution bearing estimation problem using a linear array of sensors under sensor gain uncertainties and shows better performance than MUSIC [Schmidt86] in terms of the estimated means and root mean square errors of the bearing angle estimates.

ACKNOWLEDGEMENT

This work is supported, in part, by the Office of Naval Research under contract N00014-89-J-1788 and in part by the Center for Signal and Imaging Processing with Medical Applications.

REFERENCES

[Abeles77] M. Abeles and M H. Goldstein, Jr., "Multispike Train Analysis," *IEEE Proc.* pp. 762-773, May, 1977.

[Anderson84] T. W. Anderson, *An Introduction of Multivariate Statistical Analysis,* John Wiley & Sons, 1984.

[Cardoso89] J. Cardoso, "Source Separation Using Higher Order Moments", *Proc. IEEE ICASSP*, vol. 4, pp. 2109-2112, 1989.

[Friedlander88] B. Friedlander, and A. J. Weiss, "Eigenstructure Methods for Direction Finding with Sensor Gain and Phase Uncertainties," *Proc. IEEE ICASSP'88* New York, NY., pp. 2681-2684

[Gerstein83] G. L. Gerstein, M. J. Bloom, I. E. Espinosa, S. Evanczuk and M. R. Turner, "Design of a Laboratory for Multineuron Studies," *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-13, No. 5, pp. 668-676, 1983.

[Herault86] J. Herault and C. Jutten, "Space or Time Adaptive Signal Processing by Neural Network Models", *AIP Conf. Proc.*, pp. 206-211, Snowbird, UT, 1986.

[Kay88] S. M. Kay, *Modern Spectrum Estimation: Theory and Application,* Prentice Hall, Englewood Cliffs, NJ. 1988.

[Konstantinides88] K. Konstantinides, and K. Yao, "Statistical Analysis of Effective Singular Values in Matrix Rank Determination," *IEEE Trans. ASSP.*, vol. 36, pp.757-763, May, 1988.

[Kumaresan83] R. Kumaresan and D. Tufts, "Estimation of arrival of multiple plane waves," *IEEE Trans. Aerospace Electronic Systems*, vol. AES-19, pp.123-133, 1983

[Paulraj85] A. Paulraj and T. Kailath, "Direction of Arrival Estimation By Eigenstructure Methods with Unknown Sensor Gain and Phase," *Proc. IEEE ICASSP'85*, Tampa, FL., pp. 640-643, 1985.

[Pisarenko73] V. F. Pisarenko, "The Retrieval of Harmonics from Covariance," *Geophys. J. Roy. Astron. Soc.*, Vol. 33, pp. 347-366, 1973.

[Salgonicoff88] M. Salganicoff, M. Sarna, L. Sax and G. L. Gerstein, "Unsupervised waveform classification for multineuron recordings: a real time, software-based system" *J. of Neurosicience Methods*, 25, pp. 181-196, 1988.

[Schmidt84] E. M. Schmidt, "Instruments for sorting neuroelectric data: a review" *J. of Neurosicience Methods*, 12, pp. 1-24, 1984.

[Schmidt86] R. O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Trans. on Antennas and Propagation*, vol. AP-34, No. 3, pp. 276-280. March 1986.

[Vittoz] E. A. Vittoz and X. Arreguit, "CMOS Integration of Herault-Jutten Cells for Separation of Sources," to appear as a chapter of "Analog Implementations of Neural Systems," C. Mead and M. Ismail, editors, Kluwer Academic Publishers, Norwell, MA.

	MUSIC		**EFOBI**	
SNR	Angle 1	Angle 2	Angle 1	Angle 2
10dB	22.0050	33.1240	25.4111	30.6490
30dB	22.0242	33.1986	24.9640	30.1509
60dB	21.9892	33.2232	24.9543	30.1367

Table 1.a. Estimated means of Bearing Angle Estimates

	MUSIC		EFOBI	
SNR	Angle 1	Angle 2	Angle 1	Angle 2
10dB	3.0034	3.1356	1.0508	1.5625
30dB	2.9762	3.1990	0.5785	0.7818
60dB	3.0108	3.2232	0.5663	0.7687

Table 1.b Estimated RMSE of Bearing Angle Estimates

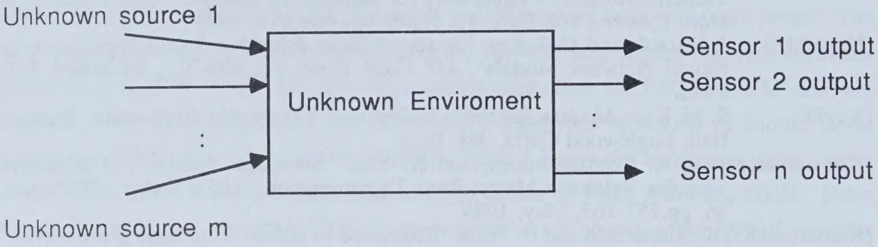

Unknown source 1
Unknown Enviroment
Sensor 1 output
Sensor 2 output
:
:
Sensor n output
Unknown source m

Figure 1: Schematic Diagram of Multiple Source Signals Separation Problem.

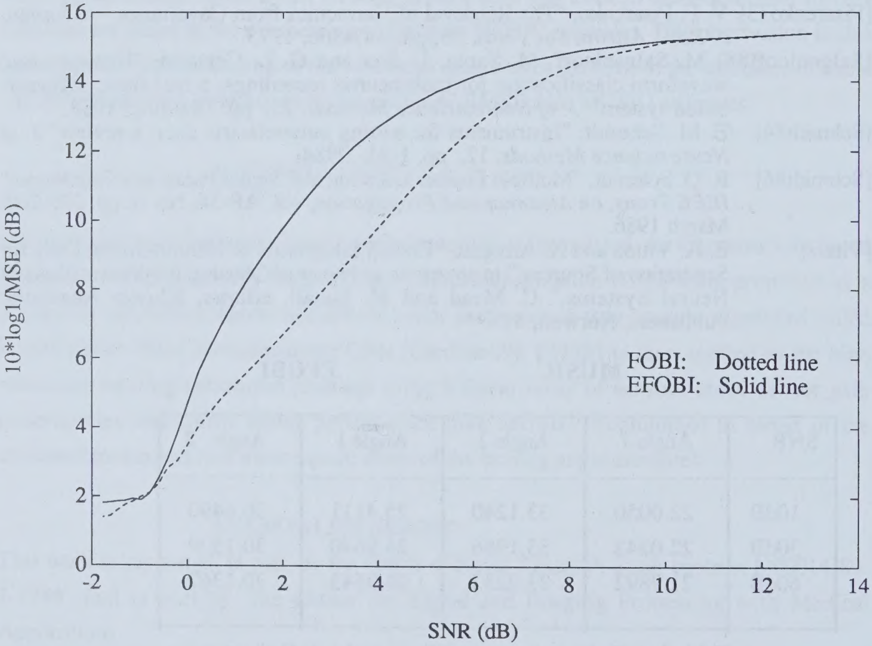


Figure 2: Comparison of EFOBI with FOBI.

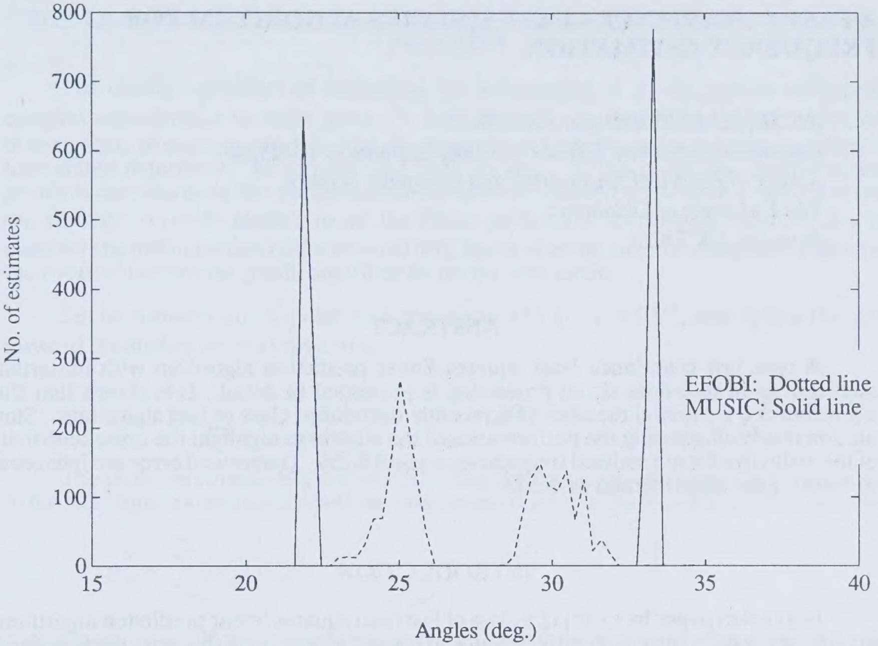


Figure 3: Histogram of Bearing Estimates with 1000 Trials at 30dB SNR.

A FAST COVARIANCE LEAST SQUARES ALGORITHM FOR FREQUENCY ESTIMATION

RICHARD M. TODD and J.R. CRUZ
Communications and Signal Processing Laboratory (CSPLab)
School of Electrical Engineering and Computer Science
The University of Oklahoma
Norman, OK 73019

ABSTRACT

A new fast covariance least squares linear prediction algorithm with important applications in real-time signal processing is presented in detail. It is shown that this algorithm is a nontrivial member of a recently introduced class of fast algorithms. Simulation results illustrating the performance of the algorithm highlight the noise sensitivity of the estimates for normalized frequencies around 0.250. Theoretical error analysis seem to confirm the experimental evidence.

INTRODUCTION

In a recent paper by Cruz [1], a class of fast least squares linear prediction algorithms for frequency estimation was introduced, and several elements of this class were derived. By formulating the least squares minimization problem in terms of an appropriate inner product, the frequency estimation problem is reduced to finding the roots of an orthogonal polynomial with respect to this inner product. Once the problem is posed in these terms, the classical work of Rutishauser [2] allows one to compute the frequency estimates directly from the data without having to root the orthogonal polynomial. This is perhaps the most significant feature of this class of algorithms, and the key to its computational efficiency.

Attempts to derive fast linear prediction algorithms which avoid the rooting of a polynomial have been previously reported in the literature. A particularly interesting one was due to Cybenko [3], who succeeded in substituting the computation of the inverse prediction filter and its roots for the computation of the eigenvalues of a tridiagonal matrix of the Lanczos constants. Although computational efficient and numerically stable, it still entailed the solution of an eigenvalue problem. Furthermore, the algorithm implemented the *autocorrelation* method of linear prediction subject to the constraint that the roots of the inverse prediction filter lie on the unit circle, and no other generalizations, such as the *covariance* method were available.

Cruz and Banjanin [4], first introduced a fast algorithm for the computation of frequency estimates using Rutishauser's work for the *autocorrelation* least squares linear prediction problem with roots constrained to lie on the unit circle, using the circulant operator formulation developed in [5]. In this paper we present a new constrained least-squares linear prediction algorithm, based on nonwindowed data, i.e. *covariance* method, for frequency estimation. This algorithm is derived using the general framework introduced in [1], and is shown to be a nontrivial element of this class of fast algorithms.

This work was supported in part by NSF grant No. RII-8610676.

THE LEAST SQUARES LINEAR PREDICTION FREQUENCY ESTIMATION PROBLEM

The classical problem of estimating the frequencies of p damped or undamped complex exponentials in noise given N data samples, has been the subject of intense investigation in recent years [6]. The exact sense in which the approximation problem is formulated determines the solution and its properties. In this paper the approximation problem considered is the minimization of the total squared error over a finite interval n_0 through n_1 with respect to *all* the linear prediction coefficients. Specifically, we consider the minimization of the forward total squared error, subject to the constraint that the roots of the inverse prediction filter be on the unit circle.

Let us consider an N-point data sequence $x[1], \ldots, x[N]$, and define the pth forward prediction error at time n as

$$e_p^f[n] = \sum_{i=0}^{p} a_p[i]x[n-i] \qquad a_p[0] = 1. \tag{1}$$

The problem considered, consists in finding the frequency estimates that correspond to the least squares minimization of the total squared error of the forward prediction errors

$$\rho_p^f = \sum_{n=n_0}^{n_1} \left|e_p^f[n]\right|^2 \tag{2}$$

with respect to *all* of the linear prediction coefficients, subject to the condition that the roots of the forward inverse prediction filter all lie on the unit circle of the complex plane.

In order to use the general mathematical framework developed in [1], let the inner product of two polynomials $F(z)$ and $G(z)$ be define as

$$< F(z), G(z) > = \sum_{i=0}^{+\infty} \sum_{j=0}^{+\infty} f^*[i] \left[\sum_{n=n_0}^{n_1} x^*[n-i]x[n-j] \right] g[j] \tag{3}$$

where

$$F(z) = \sum_{i=0}^{+\infty} f[i]z^{-i} \qquad G(z) = \sum_{i=0}^{+\infty} g[i]z^{-i}.$$

The total squared error in (2) can be expressed in terms of this inner product as

$$\rho_p^f = < A_p(z), A_p(z) > \tag{4}$$

where $A_p(z)$ is the inverse prediction filter given by

$$A_p(z) = \sum_{i=0}^{p} a_p[i]z^{-i} \qquad a_p[0] = 1 \tag{5}$$

and the minimization problem can be formulated in terms of minimizing the norm squared of the polynomial $A_p(z)$, whose roots are subject to the unit modulus constraint. It can be shown [7] that the necessary and sufficient condition for this polynomial to be of minimum norm is that

$$< A_p(z), z^{-i} > = 0 \qquad i = 1, 2, \ldots, p. \tag{6}$$

The polynomials $\{A_i(z)\}$, $i = 0, 1, 2, \ldots, p$ do not form an orthogonal set with respect to the inner product defined above, since

$$< A_i(z), A_o(z) > \neq 0 \qquad i = 1, 2 \ldots, p \tag{7}$$

however, the set of polynomials $\{z^i A_i(z)\}$ do form an orthogonal set. A three-term recursion on these polynomials exists, and can be expressed as

$$A_i(z) = A_{i-1}(z) - \alpha_i z^{-1} A_{i-1}(z) - \beta_{i-1} z^{-2} A_{i-2}(z) \quad \alpha_o = \beta_o = 0 \tag{8}$$

where

$$\alpha_i = \frac{< A_{i-1}(z), z^{-1} A_{i-1}(z) >}{< A_{i-1}(z), A_{i-1}(z) >}$$

and

$$\beta_i = \frac{< A_{i-1}(z), z^{-2} A_{i-2}(z) >}{< A_{i-2}(z), A_{i-2}(z) >}$$

with $A_o(z) = 1$ and $A_{-1}(z) = 0$.

Following the work of Rutishauser [2], let us define two parameters q_i, e_i such that

$$q_i + e_{i-1} = \alpha_i \qquad q_i e_i = \beta_i \qquad (q_o = e_o = 0) \tag{9}$$

and split the original recursion into two two-term recursions

$$A_i(z) = A^1_{i-1}(z) - q_i z^{-1} A_{i-1}(z) \tag{10}$$

$$A^1_{i-1}(z) = A_{i-1}(z) - e_{i-1} z^{-1} A^1_{i-2}(z) \tag{11}$$

where $A^1_o(z) = 1$.

Next we omit the proof that $\{z^i A^1_i(z)\}$ form an orthogonal set of polynomials with respect to the inner product $< zF(z), G(z) >$, and therefore, the polynomials $A^1_i(z)$ satisfy a triple recursion analogous to (8). Similarly, this triple recursion can be split into two two-term recursions involving polynomials $A^2_i(z)$. The set $\{z^i A^2_i(z)\}$ can be shown to be orthogonal with respect to the inner product $< z^2 F(z), G(z) >$. In general, it can be stated that a family of sets of polynomials $\{A^j_i(z)\}$, $j = 0, 1, \ldots, p$, can be generated, where each set $\{z^i A^j_i(z)\}$ is orthogonal with respect to the inner product $< z^j F(z), G(z) >$, and the coefficients associated with the recursion for each set $\{A^j_i(z)\}$ satisfy the following "rhombus rules"

$$q^j_i + e^j_{i-1} = q^{j-1}_i + e^{j-1}_i \qquad q^j_{i-1} e^j_{i-1} = q^{j-1}_i e^{j-1}_{i-1}. \tag{12}$$

It follows then, that any of these recursions will be uniquely defined, once the initial conditions for the rhombus rules are determined. One possible set of initializations, is given by the sequence

$$q_1, q_1^1, q_1^2, \ldots, q_1^j, \ldots \tag{13}$$

These coefficients can be computed from the orthogonality relations developed above. The polynomials $A_1^j(z)$ satisfy the recursion

$$A_1^j(z) = A_o^{j+1}(z) - q_1^j z^{-1} A_o^j(z) \tag{14}$$

and in addition

$$< z^j A_1^j(z), z^{-1} >= 0. \tag{15}$$

Then the q_1^j's can be determined by assuring that the previous orthogonality condition holds, i.e.

$$q_1^j = \frac{< z^j A_o^{j+1}(z), z^{-1} >}{< z^{j-1} A_o^j(z), z^{-1} >} \tag{16}$$

which can be simplified to

$$q_1^j = \frac{< 1, z^{-(j+1)} >}{< 1, z^{-j} >}. \tag{17}$$

In order to impose the condition that the roots of $A_p(z)$ must lie on the unit circle of the complex plane, let us consider the case when the data is real and p is even, say $p = 2p'$.

The polynomial $A_p(z)$ with roots constrained to be on the unit circle can be written as

$$A_p(z) = K z^{-p} \prod_{i=1}^{p} (z - z_i) \tag{18}$$

where

$$z_i = \frac{y_i}{2} + j\sqrt{1 - \frac{y_i^2}{4}} \qquad i = 1, \ldots, p' \tag{19}$$

and

$$z_{p'+1} = z_1^*, \ldots, z_p = z_{p'}^*. \tag{20}$$

Since the y_i's are real and belong to the interval [-2,2], for every $A_p(z)$ with roots on $|z| = 1$, we can find a polynomial $B_{p'}(z)$ of p'th order, with real roots in [-2,2], such that

$$A_p(z) = z^{-p'} B_{p'}\left(z + \frac{1}{z}\right). \tag{21}$$

Furthermore, $A_p(z)$ can be expressed in terms of the coefficients of $B_{p'}(z)$ as

$$A_p(z) = \sum_{i=0}^{p'} b_i \sum_{j=0}^{i} \binom{i}{j} z^{2j-i-p'} \tag{22}$$

and the total squared error in (4) can be expressed in terms of the coefficients of $B_{p'}(z)$ as

$$< A_p(z), A_p(z) >= \sum_{i=0}^{p'} \sum_{k=0}^{p'} b_{p'}[i] b^*_{p'}[k] m_{p'}[i,k] \tag{23}$$

where

$$m_{p'}[i,k] = \sum_{j=0}^{i} \sum_{l=0}^{k} \binom{i}{j} \binom{k}{l} r_p[p'+k-2l, p'+i-2j] \tag{24}$$

and

$$r_p[i,j] = \sum_{n=n_0}^{n_1} x^*[n-i] x[n-j]. \tag{25}$$

If we define a new inner product between two polynomials of degree p' as

$$< F(z), G(z) >= \sum_{i=0}^{p'} \sum_{k=0}^{p'} f^*[k] \sum_{j=0}^{i} \sum_{l=0}^{k} \binom{i}{j} \binom{k}{l} r_p[p'+k-2l, p'+i-2j] g[i] \tag{26}$$

then the total squared error can be written in terms of this inner product as

$$\rho_p^f = < B_{p'}(z), B_{p'}(z) > \tag{27}$$

and the constrained minimization problem can be formulated as the unconstrained minimization of the norm squared of the polynomial $B_{p'}(z)$.

A similar development to the one presented above for the case of the unconstrained minimization of the norm squared of $A_p(z)$ can be derived, leading to an initialization sequence formally identical to (17), in terms of the newly defined inner product given in (28). This sequence can be computed directly from the data as

$$q_1^j = \frac{\sum_{i=0}^{j+1} \binom{j+1}{i} r_p[p', p'+j+1-2i]}{\sum_{i=0}^{j} \binom{j}{i} r_p[p', p'+j-2i]}. \tag{28}$$

A FAST COVARIANCE ALGORITHM

In view of the formulation presented above, the frequency estimation problem can be posed as one of finding the roots of an orthogonal polynomial of degree p with respect to an inner product. The solution to this problem can be found directly from the data by means of a classical algorithm due to Rutishauser [2]. This algorithm is a technique for constructing orthogonal polynomials $P_i^j(z)$ with respect to an inner product $< z^j F(z), G(z) >$. This is done by computing the recurrence coeficients q_i^j, e_i^j by means of the rhombus rules given in (12). The initial conditions for these rules are the coefficients q_1^j which can be computed directly from the data, using the particular inner product in question.

The algorithm presented here stems directly from two important results due to Stiefel [8], and Rutishauser [2]. The first states that for an orthogonal polynomial of degree p, the coefficients $e_p^j, j = 0, 1, \ldots$ vanish identically. The second result establishes the convergence of the coefficients $q_i^j, j = 0, 1, \ldots$ to the ith root of the orthogonal polynomial of degree p.

By forcing the sequence $e_{p'}^j, j = 0, 1, \ldots$ to vanish identically, one can find estimates of the frequencies from the arccosines of one half the roots of the polynomial $B_{p'}(z)$.

It is important to point out the role of the range over which the total squared error is to be minimized. If we define the range of the inner product as $n_o = p + 1$ and $n_1 = N$,, this corresponds to the least squares *covariance* method of linear prediction, i.e., only available data samples are used.

EXPERIMENTAL RESULTS AND CONCLUSIONS

The first set of experimental results consists of two equal amplitude sinusoids in additive white Gaussian noise with normalized frequencies $f_1=0.025$ and $f_2=0.450$, for signal-to-noise ratios (SNR) of 10, 15, 20, and 30 dB. A 512-point data sequence was used to generate 100 estimates of the two frequencies shown in Fig. 1. A third order prediction algorithm ($p'=3$) was used. These two frequencies are widely separated, and far away from one half the Nyquist frequency. The estimates are unbiased and exhibit a very small mean square error (MSE) at high SNRs. As the SNR decreases the estimates show a significant bias as expected for any linear prediction algorithm.

TABLE I

ESTIMATION ERRORS FOR $f_1=0.025$ AND $f_2=0.450$

	SNR			
	30 dB	20 dB	15 dB	10 dB
$Bias$	0.00037045 -0.00018957	0.00239277 -0.00121371	0.00662904 -0.00353352	0.01669860 -0.00983709
$\sqrt{MSE}$	0.000374009 0.000191758	0.002404313 0.001220320	0.006652969 0.003548070	0.016745477 0.009872006

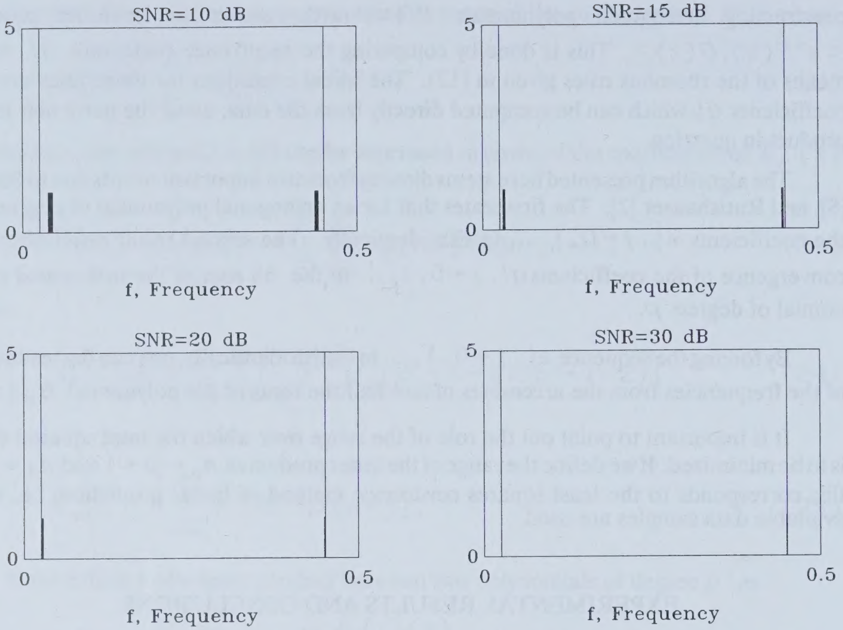


Fig. 1. Frequency estimates for 100 trials of 512 samples of two sinusoids ($f_1=0.025$ and $f_2=0.450$) in additive white Gaussian noise.

In the second set of results the sinusoid at frequency f_2 was brought to 0.300, i.e., closer to half the Nyquist frequency. All other parameters remained unchanged from the previous example. It is interesting to note that the estimate for f_2 was degraded significantly while the estimate for f_1 exhibits a larger bias. It should be pointed out that this degradation cannot be attributed to the close spacing of the two sinusoids since they are still well separated.

TABLE II

ESTIMATION ERRORS FOR $f_1=0.025$ AND $f_2=0.300$

	SNR			
	20 dB	18 dB	15 dB	10 dB
Bias	0.02930620 -0.00182874	0.02981820 0.00177890	0.03116750 -0.00586142	0.03636410 -0.00645988
$\sqrt{MSE}$	0.02931639 0.10588201	0.03130583 0.09893144	0.03119595 0.10372078	0.03642883 0.09612809

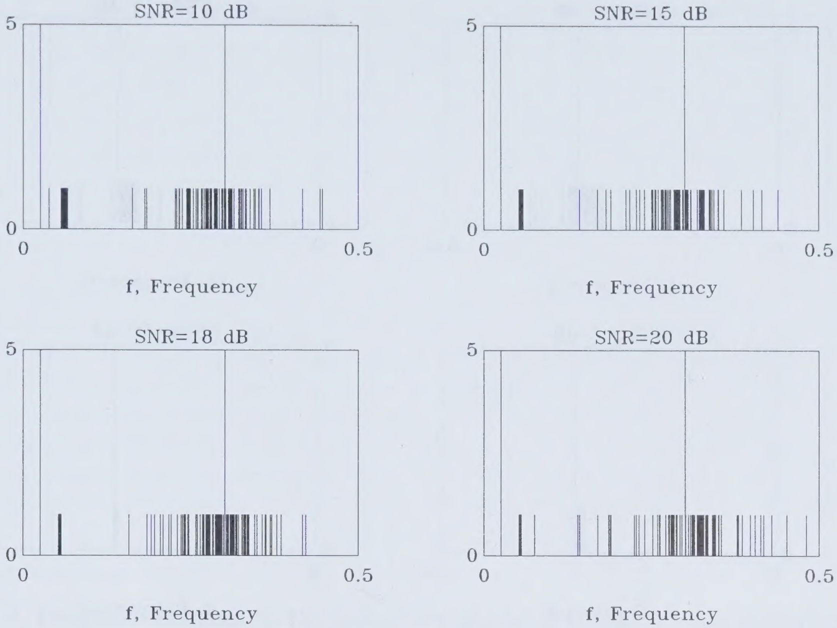


Fig. 2. Frequency estimates for 100 trials of 512 samples of two sinusoids ($f_1=0.025$ and $f_2=0.300$) in additive white Gaussian noise.

In order to further investigate the source of the degradation observed in the previous example let us consider the case of a single sinusoid, i.e., remove the sinusoid at frequency f_1. The results of this experiment using a third order prediction algorithm ($p'=3$) are shown in Fig. 3. Although the estimates are somewhat improved from the previous example, a significant deterioration of the MSE is observed particularly for low SNRs.

TABLE III

ESTIMATION ERRORS FOR $f_2=0.300$

	SNR 20 dB	 18 dB	 15 dB	 10 dB
$Bias$	-0.00320040	-0.00597677	-0.01481930	-0.01309370
$\sqrt{MSE}$	0.00332824	0.00635802	0.03843904	0.03918444

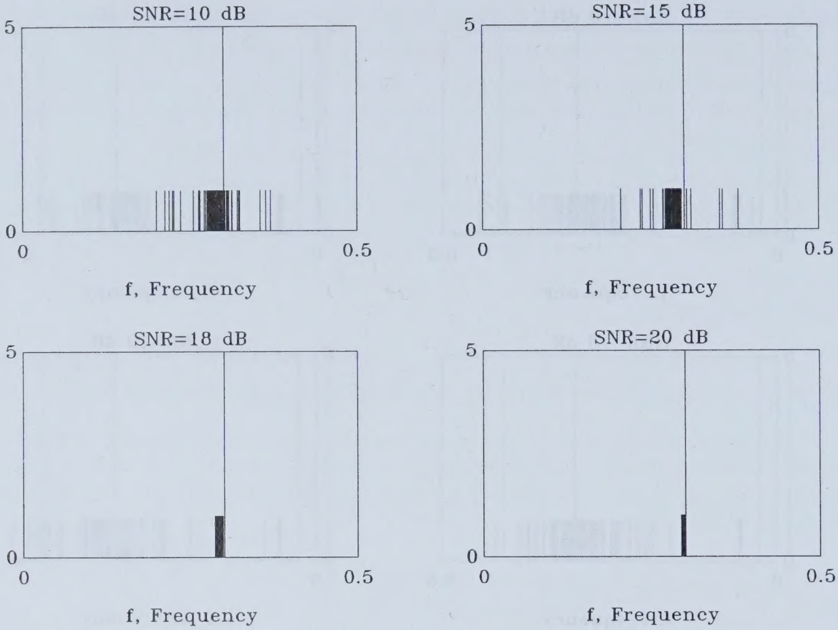


Fig. 3. Frequency estimates for 100 trials of 512 samples of a single sinusoid ($f_2 = 0.300$) in additive white Gaussian noise.

The last set of experimental results is identical to the previous one except for the frequency of the sinusoid which was set at $f_1 = 0.025$. It is quite interesting to observe from the results shown in Fig. 4 that the frequency estimates for this example do not exhibit the degradation previously observed for $f_2 = 0.300$. This is further evidence that the large MSE associated with the estimate of f_2 in both Figs. 2 and 3 is due to its value, i.e., close to the half Nyquist frequency and not to the proximity of the second sinusoid in Fig. 2.

TABLE IV

ESTIMATION ERRORS FOR $f_1 = 0.025$

	SNR			
	20 dB	18 dB	15 dB	10 dB
Bias	0.00457554	0.00686053	0.01207110	0.02636060
$\sqrt{MSE}$	0.00459805	0.00689172	0.01211759	0.02643195

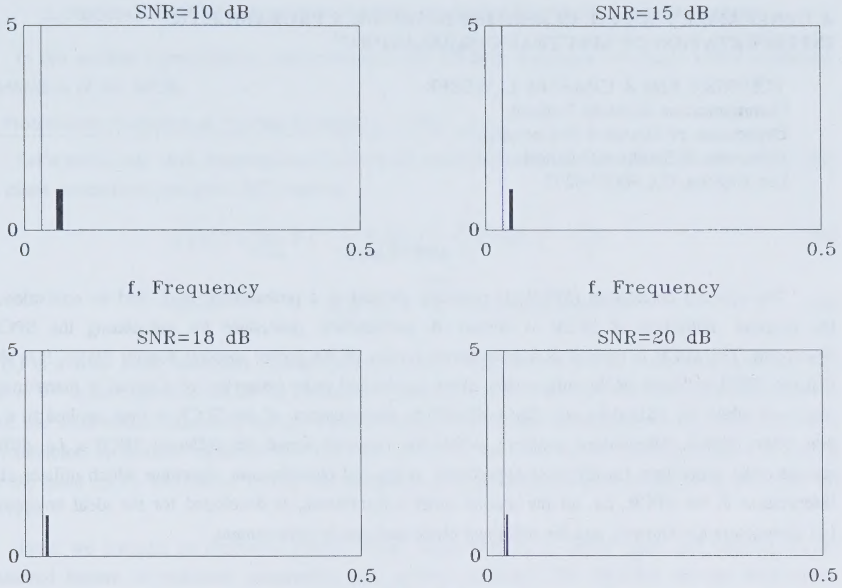


Fig. 4. Frequency estimates for 100 trials of 512 samples of a single sinusoid ($f_1 = 0.025$) in additive white Gaussian noise.

Prompted by these and other similar experimental observations we have investigated the theoretical performance of this algorithm for the simple case of a single sinusoid in white noise. It was found that the error associated with q_1^j is a function of $Q = 2\cos(2\pi f)$. For $Q \ll 1$ the error increases rapidly for a given SNR. This corresponds to a sinusoidal component close to one half the Nyquist frequency, i.e., f=0.250. This sensitivity is directly related to the mapping of orthogonal polynomials $A_p(z)$ with roots on the unit circle to orthogonal polynomials $B_{p'}(z)$ with real roots on [-2,2]. The details of the error analysis will be presented in a forthcoming paper.

REFERENCES

[1] J.R. Cruz, "A General Framework for a Class of Fast Algorithms," in *Proc. Advances in Communications and Control Systems*, (Baton Rouge, LA), Vol. I, 1988, pp. 569-575.
[2] H. Rutishauser, *Der Q-D Algorithmus*, Birkhauser: Basel, 1957.
[3] G. Cybenko, "Fast Approximation of Dominant Harmonics," *SIAM J. Sci. Stat. Comput.,* Vol. 5, No. 2, pp. 317-331, June 1984.
[4] J.R. Cruz and Z. Banjanin, "Applications of the Quotient-Difference Algorithm to Modern Spectral Estimation," in *Linear Algebra in Signals, Systems and Control*, B.N. Datta, Ed., SIAM: Philadelphia, PA, 1988.
[5] G. Cybenko, "Fast Approximation of Dominant Harmonics," *SIAM J. Sci. Stat. Comput.,* Vol. 5, No. 2, pp. 317-331, June 1984.
[6] S.M. Kay, *Modern Spectral Estimation*, Prentice Hall: Englewood Cliffs, New Jersey, 1988.
[7] J.D. Markel and A.H. Gray,Jr., *Linear Prediction of Speech*, Springer-Verlag: Berlin, 1976.
[8] E.L. Stiefel, "Kernel Polynomials in Linear Algebra and Their Applications," *Nat. Bur. Stand.-Appl. Math. Series*, 49, pp. 1-22, 1958.

A GENERALIZED SIGNAL CLASSIFIER BASED ON A PROBABILISTIC INTERPRETATION OF SPECTRAL CORRELATION [a]

YOUNGKY KIM & CHARLES L. WEBER
Communication Sciences Institute,
Department of Electrical Engineering,
University of Southern California,
Los Angeles, CA 90089-0272

ABSTRACT

The spectral correlation (SPCR) is precisely defined in a probabilistic way, and its equivalence to the original definition of SPCR is shown. A probabilistic procedure for calculating the SPCR is developed. The SPCR is viewed as a generalized version of the power spectral density (PSD). It is shown that the SPCR contains all the information about the second order properties of a signal in many interesting cases while the PSD does not. The probabilistic interpretation of the SPCR is then applied to a general binary signal classification problem, where the received signal has different SPCR's, *i.e.* different second order properties, for different hypotheses. A general classification algorithm which utilizes all the information in the SPCR, *i.e.* all the second order informations, is developed for the ideal environment (all parameters are known), and the unknown phase and epoch environment.

1. INTRODUCTION

Recently, a more general type of power spectral density, called *spectral correlation* (SPCR) has been investigated. The SPCR domain includes the power spectral density (PSD). Concepts and properties of SPCR are developed in [1], and actual spectral correlations of certain modulated signals are calculated in [2-3]. Applications of the SPCR to detection and estimation problems are considered in [4]. These references show potential advantages of the SPCR over the PSD as a signal analysis tool, namely : *1) many signals having the same/similar power spectral densities are quite different in the spectral correlation domain; (for example, BPSK and QPSK have different spectral correlations); and 2) the presence of AWGN does not affect the SPCR's of signals* [b].

Although there is ample encouragement for the utilization of SPCR as a signal processing algorithm, there is some lack of understanding in its interpretation. This is due primarily to its non-probabilistic definition. With a probabilistic interpretation or an equivalent probabilistic definition, the SPCR could be analytically as well as empirically compared to alternative methods/algorithms. In the next section, we present a probabilistic definition of the SPCR, show its (mean square sense) equivalence to the original definition of SPCR, and develop a probabilistic procedure for calculating SPCR. The equivalent probabilistic definition of SPCR clarifies that the SPCR is a generalized version of the ensemble power spectral density (PSD), and that it contains all the second order information about a signal in many interesting cases, while the PSD does not.

In the latter part of this paper, we consider an application of this probabilistic interpretation of the SPCR to a binary signal classification problem, where the received signal has different SPCR's, *i.e.* different second order statistics, for different hypotheses. The problem will be interpreted as a binary hypothesis testing problem. A mathematical model is formulated and a general classifier, called *the SPCR Classifier*, is developed.

---

[a] Acknowledgment : This work was supported in part by ARO Contract #DAAL03-88-K-0037, and in part by TRW Contract #23613R18S.

[b] This is true because the definition of SPCR includes a doubly infinite time observation. The practical utilization of SPCR clearly involves a finite time observation, wherein the presence of AWGN plays a significant role.

2. PROBABILISTIC INTERPRETATION OF THE SPECTRAL CORRELATION

In this section, a probabilistic interpretation of the SPCR is developed. We begin with a probabilistic definition of the SPCR.

2.1. Probabilistic Definition of Spectral Correlation : $S_x^{\alpha}(f\ ;\underline{p}\ )$

Let's tentatively (and inappropriately) define the ensemble spectral correlation of a real signal $x(t)$ as a direct generalized ensemble PSD, namely

$$S'^{\alpha}_{x}(f) \equiv \lim_{T\to\infty} E\{\ \frac{1}{T} X_T(0, f+\frac{\alpha}{2})\, X_T^*(0, f-\frac{\alpha}{2})\ \} \tag{1a}$$

$$\text{where} \qquad X_T(t, s) = \int_{t-T/2}^{t+T/2} x(v)\ \ exp(-2\pi s v)\, dv \tag{1b}$$

If $x(t)$ is a wide sense stationary random process, the SPCR, defined above, is equal to zero for $\alpha \neq 0$ since $e^{j2\pi st}$ is an eigenfunction of the covariance function of $x(t)$ with eigenvalue $S'^{0}_{x}(s)$. But, in many cases, the wide sense stationarity comes from averaging over certain parameters of $x(t)$ after they have been modeled as random variables. For example, BPSK is not wide sense stationary before averaging over the baud epoch and carrier phase. The empirical average avoids this ensemble average in (1a) [1] (see (7)).

Here, we consider an ensemble average where variables such as carrier phase and baud epoch are considered known or unknown parameters, not random variables. The resulting process is a cyclo-stationary process [1,5].

In order to define an *Ensemble Spectral Correlation*, we consider a random process $x(t)$, for which a parameter vector $\underline{p}$ can be defined as follows :

$$\underline{p}\ :\ \left\{ \begin{array}{l} \textit{set of parameters}\ ;\ \text{for}\ \textit{fixed}\ \underline{p},\ \textit{there exists a finite}\ T_0 \\ \textit{such that}\ x(t_1;\underline{v},\underline{p})\ \text{and}\ x(t_2;\underline{v},\underline{p})\ \textit{are independent}\ \ \text{if}\ |t_1-t_2| > T_0 \end{array} \right. \tag{2}$$

where $\underline{v}$ is the probability space. For example, $\underline{p}$ is the baud epoch and carrier phase for MPSK, and $\underline{p}$ is the baud epoch for noncoherent MFSK. For some random processes like white gaussian noise, $\underline{p}$ is empty. For the fixed $\underline{p}$, $x(t;\underline{v},\underline{p})$ may not be a wide sense stationary random process, but alternatively a cyclo-stationary process. And, the expectation in (1), for the fixed $\underline{p}$, may be non-zero for $\alpha \neq 0$.

We now define this conditional expectation, the *Ensemble Spectral Correlation*, $S_x^{\alpha}(f\ ;\underline{p})$, as

$$S_x^{\alpha}(f\ ;\underline{p}) \equiv \lim_{T\to\infty} E_{\underline{v}}\{\ \frac{1}{T} X_T(0, f+\frac{\alpha}{2};\underline{v},\underline{p})\, X_T^*(0, f-\frac{\alpha}{2};\underline{v},\underline{p})\ \big|\ \underline{p}\ \ \} \tag{3}$$

2.2. Equivalence of Two Definition of SPCR

In this sub-section, the equivalence between the ensemble SPCR and the empirical SPCR is proven. We first develop a relation between the ensemble SPCR and the autocorrelation of a real signal $x(t;\underline{v},\underline{p}\ )$, $R_x(\ t,\tau\ ;\underline{p}\ )$, which is defined as

$$R_x(\ t,\tau;\ \underline{p}\ )\ \equiv\ E_{\underline{v}}\{\ \ x(t+\tau/2;\underline{v},\underline{p}\ )\, x(t-\tau/2;\underline{v},\underline{p}\ )\ |\ \underline{p}\ \ \} \tag{4}$$

Using the above definition and (3), the ensemble SPCR can be expressed as

$$\begin{aligned} S_x^{\alpha}(f\ ;\underline{p}) &\equiv \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}\int_{-T/2}^{T/2} E_{\underline{v}}\{\ \ x(t_1;\underline{v},\underline{p})\, x(t_2;\underline{v},\underline{p})\ |\ \underline{p}\ \ \}\ e^{-j2\pi\alpha(\frac{t_1+t_2}{2})}\, e^{-j2\pi f(t_1-t_2)}\ dt_1 dt_2 \\ &= \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}\int_{-T/2}^{T/2} R_x(\frac{t_1+t_2}{2}, t_1-t_2, ;\underline{p}\ )\ e^{-j2\pi\alpha(\frac{t_1+t_2}{2})}\, e^{-j2\pi f(t_1-t_2)}\ dt_1 dt_2 \end{aligned} \tag{5}$$

With the substitutions $\{ (t_1+t_2)/2 \rightarrow t,\ (t_1-t_2) \rightarrow \tau \}$, (5) can be expressed as

$$S_x^\alpha(f\ ;\underline{p}) = \lim_{T\to\infty} \int_{-T}^{T} \left[\ \frac{1}{T} \int_{\frac{-T+|\tau|}{2}}^{\frac{T-|\tau|}{2}} R_x(t,\tau\ ;\underline{p}\ )\, e^{-j2\pi\alpha t}\, dt\ \right] e^{-j2\pi f\tau}\, d\tau \tag{6}$$

We now assume $E[\ x(t)\ |\ \underline{p}\ ] = 0$ (a corresponding statement can similarly be developed for $x(t)$ having non-zero ensemble average). Then, since $x(t)$ and $x(t+\tau)$ are assumed independent for $|\tau| > T_0$ in (2), the autocorrelation, $R_x(t,\tau\ ;\underline{p})$ is zero for $|\tau| > T_0$ and (6) can be expressed as

$$S_x^\alpha(f\ ;\underline{p}) = \int_{-\infty}^{\infty} \left[\ \lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{T}{2}} R_x(t,\tau\ ;\underline{p}\ )\, e^{-j2\pi\alpha t}\, dt\ \right] e^{-j2\pi f\tau}\, d\tau \tag{7}$$

Now, let's consider the empirical SPCR, $\hat{S}_x^\alpha(f)$, which was defined in [1] as

$$\hat{S}_x^\alpha(f) \equiv \lim_{T\to\infty} \lim_{\delta\to\infty} \frac{1}{\delta} \int_{-\delta/2}^{\delta/2} \frac{1}{T} X_T(t, f+\frac{\alpha}{2})\, X_T^*(t, f-\frac{\alpha}{2})\, dt \tag{8}$$

In [1], it is shown that the empirical SPCR can be expressed as

$$\hat{S}_x^\alpha(f) = \int_{-\infty}^{\infty} \hat{R}_x^\alpha(\tau)\, e^{-j2\pi f\tau}\, d\tau \tag{9a}$$

$$\textit{where} \qquad \hat{R}_x^\alpha(\tau) \equiv \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t+\frac{1}{2}\tau)\, x(t-\frac{1}{2}\tau)\, e^{-j2\pi\alpha t}\, dt \tag{9b}$$

Let $g(t)$ denote $x(t+\frac{1}{2}\tau)\, x(t-\frac{1}{2}\tau)\, e^{-j2\pi\alpha t}$. Then, since $x(t)$ and $x(t')$ are assumed independent for $|t-t'|>T_0$ at (2), $g(t)$ and $g(t')$ are independent if (t,t') does not belong to $D^2 \equiv \{ (t,t')\ |\ \ |t-t'| < T_0$ or $|t+\tau-t'| < T_0$ or $|t-\tau-t'| < T_0\ \}$. Also, it is not really restrictive to assume that $E[\ |g(t)|^2\ |\ \underline{p}\ ]$ is finite. Now, the identity developed in the Appendix can be applied to (9b), which results in

$$\hat{R}_x^\alpha(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} E[\ x(t+\frac{1}{2}\tau)\, x(t-\frac{1}{2}\tau)\, e^{-j2\pi\alpha t}\ \ |\ \underline{p}\ ]\ \ dt$$

Using (4), the above equation can be re-expressed as

$$\hat{R}_x^\alpha(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} R_x(\ t, \tau\ ;\underline{p})\, e^{-j2\pi f\tau}\ \ dt \tag{10}$$

Consequently, the empirical SPCR can be expressed as follows, which directly shows the equality between the ensemble SPCR, $S_x^\alpha(f\ ;\underline{p})$ and the empirical SPCR, $\hat{S}_x^\alpha(f)$.

$$\hat{S}_x^\alpha(f) = \int_{-\infty}^{\infty} \left[\ \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} R_x(\ t, \tau\ ;\underline{p})\, e^{-j2\pi f\tau}\ \ dt \right] e^{-j2\pi f\tau}\, d\tau\ =\ S_x^\alpha(f\ ;\underline{p}) \tag{11}$$

where the $\underline{p}$ used on the right hand side is the same as the $\underline{p}$ on the left hand side.

2.3. Calculation of Spectral Correlation

The SPCR of $x(t;\underline{v},\underline{p})$ will be expressed in terms of the SPCR of its complex envelope, $u(t;\ \underline{v},\ \underline{p}_u)$ (see c)). Any bandpass signal, $x(t;\underline{v},\underline{p})$ can be expressed as

$$x(t;\underline{v},\underline{p}) = \mathrm{Re}\left\{\ \sqrt{2P}\ \ u(t;\ \underline{v},\ \underline{p}_u)\ \exp[j(2\pi f_0 t + \theta)]\ \right\} \tag{12}$$

where P is the signal power, θ is the carrier phase and $u(t;\, \underline{v}, \underline{p})$ is the unit power complex envelope of $x(t;\, \underline{v}, \underline{p})$ which in general is complex. The $\underline{p}$ of $x(t;\, \underline{v}, \underline{p})$ usually consists of the $\underline{p}$ of $u(t;\, \underline{v}, \underline{p}_u)$ plus the carrier phase, θ. Hereafter, the $\underline{p}$ of $u(t;\, \underline{v}, \underline{p}_u)$ is denoted by $\underline{p}_u$ c). First, let's define two kinds of SPCR of the complex envelope $u(t;\, \underline{v}, \underline{p}_u)$ as

$$S_{uu^*}^{\alpha}(f;\underline{p}_u) \equiv \lim_{T\to\infty} \frac{1}{T} E_{\underline{v}}\{\, [\int_{-T/2}^{T/2} u(t_1;\, \underline{v}, \underline{p}_u) e^{-j2\pi(f+\frac{\alpha}{2})t_1} dt_1]\, [\int_{-T/2}^{T/2} u(t_2;\, \underline{v}, \underline{p}_u) e^{-j2\pi(f-\frac{\alpha}{2})t_2} dt_2]^* \mid \underline{p}_u \,\} \tag{13a}$$

$$S_{uu}^{\alpha}(f;\underline{p}_u) \equiv \lim_{T\to\infty} \frac{1}{T} E_{\underline{v}}\{\, [\int_{-T/2}^{T/2} u(t_1;\, \underline{v}, \underline{p}_u) e^{-j2\pi(f+\frac{\alpha}{2})t_1} dt_1]\, [\int_{-T/2}^{T/2} u^*(t_2;\, \underline{v}, \underline{p}_u) e^{-j2\pi(f-\frac{\alpha}{2})t_2} dt_2]^* \mid \underline{p}_u \,\} \tag{13b}$$

With these definitions, the SPCR of $x(t;\, \underline{v}, \underline{p})$ can be expressed as

$$S_x^{\alpha}(f;\underline{p}) = \frac{P}{2}[\, S_{uu^*}^{\alpha}(f-f_0;\, \underline{p}_u) + S_{uu^*}^{\alpha}(-f-f_0;\, \underline{p}_u) + S_{uu}^{\alpha-2f_0}(f;\, \underline{p}_u) e^{j2\theta} + [\, S_{uu}^{-\alpha-2f_0}(f;\, \underline{p}_u) e^{j2\theta}\,]^* \,] \tag{14}$$

As shown above, the SPCR's of $u(t;\, \underline{v}, \underline{p}_u)$ completely describes the SPCR of $x(t;\, \underline{v}, \underline{p})$. The remaining problem is to calculate the SPCR's of $u(t;\, \underline{v}, \underline{p}_u)$. In order to calculate them, we define the following autocorrelations of the complex envelope $u(t;\, \underline{v}, \underline{p}_u)$ as

$$R_{uu^*}(t,\tau;\, \underline{p}_u) \equiv E_{\underline{v}}\{\, u(t+\tau/2;\, \underline{v}, \underline{p}_u)\, u^*(t-\tau/2;\, \underline{v}, \underline{p}_u) \mid \underline{p}_u \,\} \tag{15a}$$

$$R_{uu}(t,\tau;\, \underline{p}_u) \equiv E_{\underline{v}}\{\, u(t+\tau/2;\, \underline{v}, \underline{p}_u)\, u(t-\tau/2;\, \underline{v}, \underline{p}_u) \mid \underline{p}_u \,\} \tag{15b}$$

In general, these autocorrelations can be expressed in terms of periodic components ($\{a_k\}$, $\{b_j\}$) and non-periodic components ($g(\cdot)$, $h(\cdot)$) as

$$R_{uu^*}(t,\tau;\, \underline{p}_u) = \sum_{k=1}^{L} \sum_{i=-\infty}^{\infty} a_k(t+iT_k, \tau;\, \underline{p}_u) + g(t,\tau;\, \underline{p}_u) \tag{16a}$$

$$R_{uu}(t,\tau;\, \underline{p}_u) = \sum_{j=1}^{M} \sum_{i=-\infty}^{\infty} b_j(t+iT_j, \tau;\, \underline{p}_u) + h(t,\tau;\, \underline{p}_u) \tag{16b}$$

The above expressions look very complicated. But, for many interesting cases such as MPSK and MFSK, they reduces to trivial ones, i.e. $L = M = 1$ and $g(\cdot) = h(\cdot) = 0$. Using (16) with (13), the SPCR's of $u(t;\, \underline{v}, \underline{p}_u)$ can be easily shown to be

$$S_{uu^*}^{\alpha}(f;\, \underline{p}_u) = \begin{cases} = 2D\!-\!F\{\, \frac{1}{T_k} a_k(t,\tau;\, \underline{p}_u) \,\} & \text{if } \alpha = \frac{n}{T_k} \\ = 1D\!-\!F\{\, \hat{R}_{uu^*}(t,\tau;\, \underline{p}_u) \,\} & \text{if } \alpha = 0 \\ = 0 & \textit{otherwise} \end{cases} \tag{17}$$

where $2D\!-\!F\{\}$ is the two dimensional Fourier Transform, which converts t and τ to α and f respectively, and $\hat{R}()$ means the time average over t.

For $S_{uu}^{\alpha}(f;\, \underline{p}_u)$, the same result as the above is obtained with $a_k()$ and $g()$ replaced by $b_j()$ and $h()$ respectively. The above expression shows that the PSD (SPCR at α=0) has only the information on the average value of the autocorrelation function while the SPCR fully describes the autocorrelation function except its non-periodic components.

2.4. Concluding Remarks on the Probabilistic Interpretation of the SPCR

By defining $\underline{p}$, we generalized the PSD to the SPCR in a probabilistic way. The fact that this SPCR is defined in a probabilistic way is important, since other probabilistic concepts (like gaussianity) can be

---

c) $\underline{p}_u$ is the set of parameters ; for fixed $\underline{p}_u$, there exists finite T_o such that $u(t;\, \underline{v}, \underline{p}_u)$ and $u(t+\tau)$ are independent if $|\tau| > T_o$.

utilized together with SPCR concepts without any intermediate steps.

The equivalence between the probabilistic SPCR and the empirical SPCR can be interpreted as an cycloergodicity of a signal under fixed $\underline{p}$. In general, this cycloergodicity is not maintained without fixing $\underline{p}$.

As seen in (17), the PSD has information only about the average value of $R_{uu^*}(t,\tau\,;\underline{p}_u)$, while the SPCR has all the information about $R_{uu^*}(t,\tau\,;\underline{p}_u)$ and $R_{uu}(t,\tau\,;\underline{p}_u)$ except their non-periodic components. In other words, the two signals have the same PSD's if and only if the average value of the autocorrelation function is the same, while the two signals have the same SPCR if and only if all periodic components of the autocorrelations are the same.

A procedure for calculating the SPCR was developed. The former procedure utilized the input-output relation of the Linear Periodic Time Variant system (LPTV)[2-3] to calculate the SPCR while our procedure utilized the 2-D Fourier Transform, which is a more familiar and developed tool than the LPTV. We expressed the SPCR of $x(t;\,\underline{v},\underline{p})$ in terms of SPCR's of the complex envelope, $u(t;\,\underline{v},\,\underline{p}_u)$. In general, it is easier to calculate the SPCR of $u(t;\,\underline{v},\,\underline{p}_u)$ than to calculate $x(t;\,\underline{v},\underline{p})$ directly.

3. GENERALIZED SPCR CLASSIFIER FROM LRT

In the previous section, we developed a probabilistic interpretation of the SPCR. Based on this interpretation, a general signal classifier utilizing the SPCR domain is now developed.

3.1. System Model

The received signal, $r(t)$, consists of the desired signal $s(t;\,\underline{v},\underline{p})$ plus noise $n(t)$ as

$$r(t) = s(t;\,\underline{v},\underline{p}) + n(t) = \mathrm{Re}\{\ c(t)\ exp(\ j2\pi f_c t\ )\ \} \tag{18}$$

where $s(t;\,\underline{v},\underline{p})$ is one of a set of digitally modulated signals, $n(t)$ is white gaussian noise (WGN) with one-sided power spectral density N_0 W/Hz, $c(t)$ is the complex envelope of $r(t)$ and f_c is the carrier frequency. A general expression for $s(t;\,\underline{v},\underline{p})$ is

$$s(t;\,\underline{v},\underline{p}) = \mathrm{Re}\{\ \sqrt{2S}\ u(t;\,\underline{v},\,\underline{p}_u)\ \exp[\ j(2\pi f_c t+\theta)\ ]\ \} \tag{19}$$

where S is the signal power, θ is the carrier phase and $u(t;\,\underline{v},\,\underline{p}_u)$ is the unit magnitude complex envelope of the modulated signal. The carrier phase, θ, is assumed to be fixed during the entire observation whether known or unknown[d]. We limit our interests to digitally modulated signals whose $u(t;\,\underline{v},\,\underline{p}_u)$ can be expressed as

$$u(t;\,\underline{v},\,\underline{p}_u) = u_0(t-\varepsilon T_b;\,\underline{v}) = \sum_{i=-\infty}^{\infty} d_i\ p(t-iT_b-\varepsilon T_b) \tag{20}$$

where $u_0(t;\,\underline{v})$ is equal to the $u(t;\,\underline{v},\,\underline{p}_u)$ of $\varepsilon=0$, T_b is the baud time, ε is the normalized baud epoch time and $p(t)$ is the baud pulse function. For $\{d_i\}$, we assume that i) $E(d_i)$ is equal to zero and, ii) there exists finite I such that d_i and d_j are independent if $|i-j| > I$ (i.e. $\underline{p}_u=\{\varepsilon\}$).

Depending on the model of the carrier phase θ, and the epoch ε, various environments can take place. Among them, the ideal environment, and an unknown phase and epoch environment are considered. For an unknown phase and epoch environment, the carrier phase θ, and the epoch ε, are modeled as random variables uniformly distributed over $[0,\,2\pi]$ and $[0,\,1]$ respectively. For the ideal environment, both the phase and epoch are modeled as known constants. The observation time, T, is assumed to be much larger than the dependence interval of $u(t)$ (i.e. lT_b). Such an assumption is not really restrictive, since in practice, T is generally much larger than the baud time (T_b).

---

[d] Since θ is assumed to be fixed, the θ is one of the eligible elements of $\underline{p}$. And, this assumption means the local oscillator is ideally synchronized (i.e. frequency locked) to the carrier frequency.

The goal of this section is to develop a SPCR classifier for a binary hypothesis testing problem where the desired signal, $s(t; \underline{v}, \underline{p})$, has different SPCR's for different hypotheses[e] (i.e., the SPCR of $s(t; \underline{v}, \underline{p})$ is $S_{s_i}^{\alpha}(f; \theta, \varepsilon)$ for H_i). The developed classifier utilizes all of the information of the SPCR domain.

3.2. Development of SPCR Classifier

The SPCR classifier is developed based on the observation that the lower order expansion of the likelihood function (L.F.) can be expressed in the SPCR domain with or without any assumption on the signal. The classifier is obtained as one comparing the lower order expansion of the L.F. under different hypotheses, with a threshold after the lower order expansion which is expressed in the SPCR domain. The reason that we develop the classifier from the likelihood ratio test (LRT) is that any optimality for a binary hypothesis testing problem can come only from the LRT. The developed classifier is locally optimum, i.e. it becomes optimum as SNR goes down (the lower order approximation of the L.F. becomes closer to the exact L.F. as SNR goes to zero).

3.2.1. *Ideal Environment*

For the AWGN channel, the L.F. for a given signal (i.e. given $u_0(t)$, epoch time ε and carrier phase θ) is easily obtained as

$$l(\underline{r} \mid u_0(t), \varepsilon, \theta) = c \exp\left[\frac{\sqrt{2S}}{N_0} \int_0^T \mathrm{Re}\{ c^*(t)\, u(t; \underline{v}, \underline{p}_u)\, e^{j\theta} \}\, dt \right] \tag{21}$$

From (21), the second order expansion of L.F., $l_2(\underline{r} \mid u_0(t), \varepsilon, \theta)$, is obtained as follows using that $e^x \mid_{\substack{\textit{second order}\\ \textit{expansion}}} = 1 + x + \frac{1}{2}x^2$.

$$l_2(\underline{r} \mid u_0(t), \varepsilon, \theta) =$$

$$= c \left[1 + \frac{\sqrt{2S}}{N_0} \int_0^T \mathrm{Re}\{ c^*(t) u(t)\, e^{j\theta} \}\, dt + \frac{S}{N_0^2} \left[\int_0^T \mathrm{Re}\{ c^*(t) u(t)\, e^{j\theta} \}\, dt \right]^2 \right] \tag{22}$$

For this environment, both the carrier phase, θ and the epoch ε are known (i.e. they are equal to zero) and the only randomness comes from the random data, *i.e.* $u_0(t)$. After averaging over $u_0(t; \underline{v})$ (not over the known epoch), the second order expansion can be expressed as follows by using (15) and the fact that $E(u(t; \underline{v}, \underline{p}_u) \mid \varepsilon{=}0, \theta{=}0) = 0$.

$$l_2(\underline{r} \mid \varepsilon{=}0, \theta{=}0) = c \left[1 + \frac{S}{2N_0^2} \int_0^T\!\!\int_0^T \mathrm{Re}\{ c^*(t_1)\, c^*(t_2)\, R_{uu}\left(\frac{t_1+t_2}{2}, t_1-t_2; \varepsilon{=}0\right) \right.$$

$$\left. + c^*(t_1)\, c(t_2)\, R_{uu^*}\left(\frac{t_1+t_2}{2}, t_1-t_2; \varepsilon{=}0\right) \}\, dt_1 dt_2 \right] \tag{23}$$

With $c(t)$ replaced by $c(t) \prod((t-0.5T)/T)$ (where $\prod(\cdot)$ is a unit amplitude rectangular pulse of unit duration which is symmetric around zero), this expansion can be expressed in the SPCR domain by applying Parseval's relation as

$$l_2(\underline{r} \mid \varepsilon{=}0, \theta{=}0) =$$

---

[e] Two signals have different SPCR if any periodic component of their autocorrelations is different. It might be caused by different modulation formats, different baud times etc..

$$= c \, [\; 1 + \frac{S}{2N_0^2} \, \mathrm{Re}\{ \; \sum_{\alpha} \int_{-\infty}^{\infty} [\; [\tilde{S}_{cc}^{\alpha}(f)]^* \, S_{uu}^{\alpha}(f \, ; \varepsilon{=}0) + [\tilde{S}_{cc^*}^{\alpha}(f)]^* \, S_{uu^*}^{\alpha}(f \, ; \varepsilon{=}0) \;] \; df \; \} \;] \tag{24}$$

where $S_{uu}^{\alpha}(f)$ and $S_{uu^*}^{\alpha}(f)$ are defined by (13) and, $\tilde{S}_{cc}^{\alpha}(f)$ and $\tilde{S}_{cc^*}^{\alpha}(f)$ are defined as

$$\tilde{S}_{cc}^{\alpha}(f) \equiv [\; \int_0^T c(t_1) \exp[-j2\pi(f + \frac{\alpha}{2})t_1] \, dt_1 \;] \; [\; \int_0^T c^*(t_2) \exp[-j2\pi(f - \frac{\alpha}{2})t_2] \, dt_2 \;]^* \tag{25a}$$

$$\tilde{S}_{cc^*}^{\alpha}(f) \equiv [\; \int_0^T c(t_1) \exp[-j2\pi(f + \frac{\alpha}{2})t_1] \, dt_1 \;] \; [\; \int_0^T c(t_2) \exp[-j2\pi(f - \frac{\alpha}{2})t_2] \, dt_2 \;]^* \tag{25b}$$

By using (14) and (27) (which is directly obtained from (25)), the expansion can be shown to be equal to

$$l_2(\underline{r} \mid \varepsilon{=}\theta{=}0) = c \, [\; 1 + \frac{2}{N_0^2} \sum_{\alpha} \int_{-\infty}^{\infty} [\tilde{S}_{rr^*}^{\alpha}(f)]^* \, S_s^{\alpha}(f \, ; \varepsilon{=}\theta{=}0) \; df \;] \tag{26}$$

$$\tilde{S}_{rr^*}^{\alpha}(f) = \frac{1}{4} [\; \tilde{S}_{cc^*}^{\alpha}(f - f_0) + \tilde{S}_{cc^*}^{\alpha}(-f - f_0) + \tilde{S}_{cc}^{\alpha - 2f_0}(f) + [\, \tilde{S}_{cc}^{-\alpha - 2f_0}(f) \,]^* \;] \tag{27}$$

The right term of (26) was obtained in earlier work [4] with some hard arguments and its equivalent ones were obtained in [6,8]. They showed this term is an approximation of the likelihood ratio for the detection of a weak gaussian signal embedded in AWGN. We show that the approximation can also be obtained through the second order expansion of the L.F. Based on this expansion, an algorithm of the signal classification is readily obtained as

$$Z_2^{(1,1)} \equiv \frac{[\, 1 + \frac{2}{N_0^2} \sum_{\alpha} \int_{-\infty}^{\infty} [\tilde{S}_{rr^*}^{\alpha}(f)]^* \, S_{s_1}^{\alpha}(f \, ; \varepsilon{=}\theta{=}0) \, df \,]}{[\, 1 + \frac{2}{N_0^2} \sum_{\alpha} \int_{-\infty}^{\infty} [\tilde{S}_{rr^*}^{\alpha}(f)]^* \, S_{s_0}^{\alpha}(f \, ; \varepsilon{=}\theta{=}0) \, df \,]} \; \underset{H_0}{\overset{H_1}{\gtrless}} \; \mathit{Threshold} \tag{28}$$

where $S_{s_i}^{\alpha}(f)$ is the SPCR of $s(t\, ; \underline{v}, \underline{p})$ for H_i.

As shown in the above equation, the algorithm (classifier) is well interpreted in the SPCR domain, namely; i) measure the SPCR of the received signal ($\tilde{S}_{rr^*}^{\alpha}(f)$), ii) correlate the measured SPCR with each candidate SPCR ($S_{s_i}^{\alpha}(f)$), and iii) take the ratio of correlation values and compare it with the threshold. It utilizes all the information in the SPCR domain in a manner that is optimum as SNR goes asymptotically to zero. But, this classifier cannot be used when either the phase or the epoch is unknown since the $S_s^{\alpha}(f)$ is a function of those parameters. If incorrect values of those parameters are used in (26), the summation over α ($\sum_{\alpha}$) might be done in a destructive way rather than a constructive way [4].

3.2.2. *Unknown Phase* and *Epoch Environment*

When the phase is unknown (i.e. the phase is uniformly distributed over $[-\pi, \pi]$), the L.F. for a given $u_0(t\, ; \underline{v})$ and epoch time is directly obtained from (21) by taking an ensemble average over phase, as

$$l(\underline{r} \mid u_0(t\, ; \underline{v}), \varepsilon) = c \, I_0(\, \frac{\sqrt{2S}}{N_0} \left| \int_0^T c^*(t) \, u(t\, ; \underline{v}, \underline{p}_u) \, dt \right| \,) \tag{29}$$

where $I_0(\cdot)$ is the modified Bessel function of the first kind. From this equation, the second order expansion of the L.F., $l_2(\underline{r} \mid u_0(t\, ; \underline{v}), \varepsilon)$ is obtained as follows using that $I_0(x) |_{\mathit{second\ order\ expansion}} = 1 + x^2/4$.

$$l_2(\underline{r} \mid u_0(t\, ; \underline{v}), \varepsilon) = c \, [\; 1 + \frac{S}{2N_0^2} \left| \int_0^T c^*(t) \, u(t\, ; \underline{v}, \underline{p}_u) \, dt \right|^2 \;] \tag{30}$$

For this environment, the epoch is also unknown (i.e. the epoch ε, is uniformly distributed over [-0.5, 0.5]). After averaging over the random $u_0(t;\, \underline{v})$ and the epoch, the second order expansion (30) becomes

$$l_2(\underline{r}) = c\ [\ \ 1 + \frac{S}{2N_0^2} \int_0^T\!\!\int_0^T c^*(t_1)\, c(t_2)\ \hat{R}_{uu^*}(\frac{t_1+t_2}{2}, t_1-t_2)\ dt_1 dt_2\ \ ] \tag{31}$$

where $\hat{R}_{uu^*}(t,\tau)$ is the time average of $R_{uu^*}(t,\tau)$ over t.

Using Parseval's relation, this expansion can be expressed in the SPCR domain as

$$l_2(\underline{r}) = c\ [\ \ 1 + \frac{S}{2N_0^2} \int_{-\infty}^{\infty} [\tilde{S}^{\,0}_{cc^*}(f)]^*\ S^{\,0}_{uu^*}(f)\ df\ \ ] \tag{32}$$

This equation shows that the classifier based on the second order expansion can be used in an unknown phase and epoch environment only when candidate signals have different PSD's. It necessitates the fourth order expansion of L.F.. The fourth order expansion, $l_4(\underline{r})$, can be expressed as follows using $I_0(x)\,|_{\substack{fourth\ order\\ expansion}} = 1 + \frac{1}{4}x^2 + \frac{1}{64}x^4$.

$$\begin{aligned} l_4(\underline{r}) = c\ [\ \ 1 &+ \frac{S}{2N_0^2} \int_{-\infty}^{\infty} [\tilde{S}^{\,0}_{cc^*}(f\ ;\varepsilon)]^*\ S^{\,0}_{uu^*}(f)\ df\ \ + \\ &+ E_\varepsilon\{\ \ \frac{S^2}{16\,N_0^4} \int_0^T\!\!\int_0^T\!\!\int_0^T\!\!\int_0^T c^*(t_1)\, c(t_2)\, c^*(t_3)\, c(t_4)\ \ E_{\underline{v}}\{\ u(t_1)\, u^*(t_2)\, u(t_3)\, u^*(t_4)\ \}\ dt_1 dt_2 dt_3 dt_4\ \ \}\ ] \end{aligned} \tag{33}$$

where $E_\varepsilon(\cdot)$ is the expectation over the ε. Generally, the fourth order term cannot be expressed in the SPCR domain. Here, we re-consider our design principle. *The principle is to develop a reasonable algorithm which utilizes all the information in the SPCR domain. In another sense, it is a principle which utilizes only the SPCR information of the signal. Based on this principle, we assume that only the SPCR information is known. Then, it agrees with the maximum entropy philosophy to model the signal* $u(t;\, \underline{v},\, \underline{p}_u)$ *as a gaussian signal which has the same SPCR as the actual* $u(t;\, \underline{v},\, \underline{p}_u)$. With this gaussian modeling, the fourth order expansion can be expressed as

$$\begin{aligned} l_4(\underline{r}) = c\ [\ \ 1 &+ \frac{S}{2N_0^2} \int_{-\infty}^{\infty} [\tilde{S}^{\,0}_{cc^*}(f)]^*\ S^{\,0}_{uu^*}(f\ ;)\ df\ \ + \\ &+\ E_\varepsilon\{\ \ \frac{S^2}{8\,N_0^4}\ [\ \int_0^T\!\!\int_0^T c^*(t_1)\, c(t_2)\, R_{uu^*}(\frac{t_1+t_2}{2}, t_1-t_2\ ;\varepsilon)\ dt_1 dt_2\ ]^2\ \ \} \\ &+\ E_\varepsilon\{\ \ \frac{S^2}{16\,N_0^4}\ \left|\ \int_0^T\!\!\int_0^T c^*(t_1)\, c^*(t_2)\, R_{uu}(\frac{t_1+t_2}{2}, t_1-t_2\ ;\varepsilon)\ dt_1 dt_2\ \right|^2\ \ \}\ \ ] \end{aligned} \tag{34}$$

Utilizing Parseval's relation, the expansion can be expressed in the SPCR domain as follows, after averaging over the epoch.

$$\begin{aligned} l_4(\underline{r}) = c\ [\ \ 1 &+ \frac{S}{2N_0^2} \int_{-\infty}^{\infty} [\tilde{S}^{\,0}_{cc^*}(f)]^*\ S^{\,0}_{uu^*}(f)\ df\ \ + \\ &+ \frac{S^2}{8\,N_0^4} \sum_\alpha \left| \int_{-\infty}^{\infty} [\tilde{S}^{\,\alpha}_{cc^*}(f)]^*\ \ S^{\,\alpha}_{uu^*}(f\ ;\varepsilon)\ df\ \right|^2 + \frac{S^2}{16\,N_0^4} \sum_\alpha \left| \int_{-\infty}^{\infty} [\tilde{S}^{\,\alpha}_{cc}(f)]^*\ \ S^{\,\alpha}_{uu}(f\ ;\varepsilon)\ df\ \right|^2\ ] \\ = c\ [\ 1 &+ \frac{2}{N_0^2} \int_{-\infty}^{\infty} \tilde{S}^{\,0}_{rr^*}(f)\, S_s^0(f)\ df\ \ + \frac{2}{N_0^4} \sum_\alpha \left| \int_{-\infty}^{\infty} [\tilde{S}^{\,\alpha}_{rr^*}(f\ ;\varepsilon)]^*\ S_s^\alpha(f\ ;\theta,\varepsilon)\ df\ \right|^2\ ] \end{aligned} \tag{35}$$

The second equality is obtained from (14),(27) and holds only if the SPCR of the complex envelope, $u(t;\, \underline{v},\, \underline{p}_u)$, is negligible around $\pm 2f_c$ (i.e., the carrier frequency is much greater than the baud frequency). Based on this expansion, an algorithm of the signal classification is readily obtained as

$$Z_4^{(0,0)} \equiv l_4(\underline{r} \mid H_1) \; / \; l_4(\underline{r} \mid H_0) = \tag{36}$$

$$= \frac{[\, 1 + \frac{2}{N_0^2} \int_{-\infty}^{\infty} \tilde{S}^0_{rr^*}(f)\, S^0_{s_1}(f)\, df \; + \; \frac{2}{N_0^4} \sum_{\alpha} \mid \int_{-\infty}^{\infty} [\tilde{S}^{\alpha}_{rr^*}(f)]^* \, S^{\alpha}_{s_1}(f\,;\theta,\varepsilon)\, df \mid^2 \,]}{[\, 1 + \frac{2}{N_0^2} \int_{-\infty}^{\infty} \tilde{S}^0_{rr^*}(f)\, S^0_{s_0}(f)\, df \; + \; \frac{2}{N_0^4} \sum_{\alpha} \mid \int_{-\infty}^{\infty} [\tilde{S}^{\alpha}_{rr^*}(f)]^* \, S^{\alpha}_{s_0}(f\,;\theta,\varepsilon)\, df \mid^2 \,]} \; \underset{H_0}{\overset{H_1}{\gtrless}} \; \textit{Threshold}$$

As shown in the above equation, the algorithm utilizes all information in the SPCR domain in a way that the final test statistic ($Z_4^{(0,0)}$) becomes independent of the epoch and phase.

3.3. Concluding Remarks on the Generalized SPCR Classifier

By expressing the lower order expansions of the L.F. in terms of SPCR, we developed a SPCR classifier which can be applied to a general classification problem where the received signal has different SPCR's for different hypothesis, *i.e.* different second order properties. The classifier utilizes all the information of the SPCR domain, and is implementable in the spectral domain. Hence, the classifier is an appropriate one when we know (or want to utilize) only the SPCR of the received signal. The developed classifier is very close to the former classifiers of [4,6] when both the phase and epoch are known. But, it is different from the former classifiers when the phase and the epoch are unknown.

APPENDIX

- *Condition* : Let $g(t)$ be a random process which has following properties: 1) under fixed parameters $\underline{p}$ (for example, fixed epoch time and carrier phase), there exists finite T_0 and finite integer l such that $g(t)$ and $g(t')$ are independent if (t, t') does not belong to D^2, where $D^2 = \{ (t,t') \mid |t+d_1-t'| < T_0 \,, \cdots , |t+d_l-t'| < T_0 \}$, and 2) there exists finite P such that $E[\, |g(t)|^2 \mid \underline{p}\,] \le P$ for all t.
- *Lemma* : Under fixed parameters $\underline{p}$, the following equation holds in the mean square sense.

$$\lim_{\Delta \to \infty} \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} g(t)\, dt \;=_{m.s.}\; \lim_{\Delta \to \infty} \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} E[\, g(t) | \underline{p}\,]\, dt \;\equiv\; g_m \tag{A.1}$$

where $\underline{p}$ in R.H.S. is the same as $\underline{p}$ in L.H.S..

- *Proof* :

$$E[\; \mid \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} g(t)\, dt - g_m \mid^2 \; \mid \underline{p} \;] = \frac{1}{\Delta^2} \int_{-\Delta/2}^{\Delta/2} \int_{-\Delta/2}^{\Delta/2} E[\, g(t) g^*(t') | \underline{p}\,]\, dt\, dt' + |g_m|^2$$

$$- \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} E[\, g(t) | \underline{p}\,]\, g_m^*\, dt \; - \; \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} E[\, g^*(t') | \underline{p}\,]\, g_m\, dt'$$

$$= \frac{1}{\Delta^2} \int\!\!\int_{\{D_\Delta^2\}} E[\, g(t) g^*(t') | \underline{p}\,]\, dt\, dt' + \frac{1}{\Delta^2} \int\!\!\int_{\{R_\Delta^2 - D_\Delta^2\}} E[\, g(t) | \underline{p}\,]\, dt\; E[\, g^*(t') | \underline{p}\,]\, dt'$$

$$- \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} E[\, g(t) | \underline{p}\,]\, g_m^*\, dt \; - \; \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} E[\, g^*(t') | \underline{p}\,]\, g_m\, dt' + |g_m|^2$$

where $R_\Delta^2 = \{ (t, t') \mid -\frac{\Delta}{2} < t < \frac{\Delta}{2} \,,\; -\frac{\Delta}{2} < t' < \frac{\Delta}{2} \}$ and $D_\Delta^2 = D^2 \cap R_\Delta^2$.

But, there exists finite P such that $|E[\, g(t) g^*(t') | \underline{p}\,]| \le P$ in the region of D_Δ^2 since

$$| E[g(t)g^*(t') \mid \underline{p}] | \le E[|g(t)g^*(t')| \mid \underline{p}] \le (E[|g(t)|^2 \mid \underline{p}])^{1/2} (E[|g^*(t)|^2 \mid \underline{p}])^{1/2} \le P$$

Accordingly, the first term is

$$\lim_{\Delta\to\infty} \frac{1}{\Delta^2}\int\int_{\{D_\Delta^2\}} E[g(t)g^*(t') \mid \underline{p}]\, dt\, dt' \;\le\; \lim_{\Delta\to\infty} \frac{1}{\Delta^2}\int\int_{\{D_\Delta^2\}} P\, dt\, dt' = 0$$

The final equality comes from that $\lim_{\Delta\to\infty} [Area(D_\Delta^2) / \Delta^2] = 0.$

Following similar steps, one can show that other terms are

$$\lim_{\Delta\to\infty} \frac{1}{\Delta^2}\int\int_{\{R_\Delta^2 - D_\Delta^2\}} E[g(t) \mid \underline{p}]\, dt\; E[g(t')^* \mid \underline{p}]\, dt' = |g_m|^2$$

$$\lim_{\Delta\to\infty} \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} E[g(t) \mid \underline{p}]\, g_m^*\, dt = |g_m|^2, \quad \lim_{\Delta\to\infty} \frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} E[g^*(t') \mid \underline{p}]\, g_m\, dt' = |g_m|^2$$

In conclusion,

$$\lim_{\Delta\to\infty} E[\; |\frac{1}{\Delta}\int_{-\Delta/2}^{\Delta/2} g(t)\, dt - g_m|^2 \;\; \mid \underline{p} \;\;] = 0$$

REFERENCES

[1] W. A. Gardner, ''The Spectral Correlation Theory of Cyclostationary Time Series,'' Signal Processing, vol. 11, pp. 13-36, 1986.

[2] W. A. Gardner, ''Spectral Correlation of Modulated Signals: Part I- Analog Modulation,'' *IEEE Transactions on Communications*, vol. Com-35, No. 6, pp. 584-594, June 1987.

[3] W. A. Gardner, W. A. Brown, and C. K. Chen, ''Spectral Correlation of Modulated Signals: Part II- Digital Modulation,'' *IEEE Transactions on Communications*, vol. Com-35, No. 6, pp. 595-601, June 1987.

[4] W.A. Gardner, ''Signal Interception: A Unified Theoretical Framework for Feature Detection,'' *IEEE Transaction on Communications*, vol. Com-36, No. 8, pp. 897-906, August 1988

[5] A. Papoulis, ''Probability, Random Variables and Stochastic Processes'', 2nd edition, McGraw-Hill series, 1984

[6] P. Flandrin, ''A Time-Frequency Formulation of Optimum Detection'', *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, No. 9, pp. 1377-1384, September 1988

[7] I. S. Reed, ''On a Moment Theorem for Complex Gaussian Process,'' *IRE Transactions on Information Theory*, pp. 194-195, April 1962

[8] H.L. Van Trees, ''Detection, Estimation, and Modulation Theory : Part III'', John Wiley & Sons, Inc., 1971

ROBUST DIRECTION-OF-ARRIVAL ESTIMATION[†]

DAVID D. LEE and R. L. KASHYAP

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

We present a robust direction-of-arrival (DOA) estimation algorithm against outliers and distributional uncertainties. It employs a robustified Gaussian maximum likelihood (ML) estimator which performs almost as well as a Gaussian ML estimator in the pure Gaussian noise, and much better in the presence of outliers. The algorithm is equally capable of handling coherent signals as well as the single snapshot cases. Performances of the estimator in both the Gaussian and outlier contaminated Gaussian noise are investigated using the Cramer Rao Lower Bound (CRLB) and the variance derived from Influence Function (IF).

INTRODUCTION

The problem of estimating the directions of arrival (DOA) of radiating sources from measurements provided by a passive array of narrow band sensors is frequently encountered in radar, sonar, radio astronomy and seismology. In most cases the number of incident plane waves and their DOA's are to be estimated from incident source induced sensor signals. In this paper we will specifically consider the maximum likelihood (ML) DOA estimation problem in situations of outlier contaminated Gaussian noise.

The maximum likelihood (ML) technique was one of the first to be investigated [1]. Because of the high computational load of the multivariate nonlinear maximization problem involved, however, it did not become popular until recently. There are many suboptimal techniques with reduced computational load, but the performance of these techniques is usually inferior to that of the ML technique. As was also pointed out in [2], the inferiority is especially conspicuous in the threshold region, namely, when the signal to noise ratio (SNR) is small, or alternatively, when the number of snapshots is small. Moreover, these techniques cannot handle the case of coherent signals. This case appears, for example, in specular multipath propagation problems and, therefore, it is of great practical importance. The preprocessing spatial smoothing techniques proposed to cope with this problem remedy the situation only partially [3].

There has been a growing interest in the development of theory and application of robust methods [4,5]. Here, the term "robustness" refers to insensitivity against small unknown deviation in the underlying Gaussian noise assumption. Even a small deviation from the assumed Gaussian noise model can create havoc with Gaussian ML estimates since the Gaussian ML estimators are extremely sensitive to outliers. Previous DOA estimation schemes, which were developed and tested under the Gaussian assumption, usually fail to resolve close

---

[†] Partially supported by the Innovative Science and Technology(IST) Program of the SDIO, monitored by the Office of Naval Research under grant ONR N00014-85K-0611, the Engineering Research Center for Intelligent Manufacturing Systems funded by NSF under grant CDR-8803017.

DOA's when there are just one or two *outliers* out of one hundred observed sensor array snapshots.

One of the first robust techniques for DOA estimation was presented in [6], where the technique utilizes an eigenvector method and decentralized processing to obtain reliable estimates in the outlier contaminated Gaussian noise, and when a few of the subarray sites are malfunctioning. The contribution of this paper is to explore a more accurate and simpler way of obtaining robust DOA estimates using a Gaussian ML estimator robustified for contaminated Gaussian noise .

PROBLEM FORMULATION

Consider a linear array composed of M identical equi-spaced sensors. We assume that q coherent or incoherent narrow band sources, centered around a known frequency with wavelength λ, impinge on the array from directions $\theta_1,...,\theta_q$. Since narrow-bandness in the sensor array means that the propagation delays of the signals along the array are much smaller than the reciprocal of the bandwidth of the signals, the envelopes of the signals received by the array can be expressed by the following.

Let $x_j(t_i)$ and $u_j(t_i)$ denote the ith snapshots of the jth sensor output and noise, respectively. Then $x_j(t_i)$ can be expressed as

$$x_j(t_i) = \sum_{k=1}^{q} \alpha_k \, sin\,[\omega_k t_i + (2\pi/\lambda)(j-1)\, d \sin\theta_k] + u_j(t_i) \qquad i=1,...,N \,;\, j=1,...,M, \qquad (1)$$

where N is the total number of snapshots, M is the number of sensors in the array, d is the spacing between sensors, and θ_k's are the unknown DOA's. For simplicity, we assume that α_k's and ω_k's, the amplitudes and frequencies of the envelopes associated with the sources, are known prior to the estimation. In practice, α_k's and ω_k's can be estimated beforehand using a robust technique proposed in [7].

The vector of the received signals $X(t_i)$ can be expressed as

$$X(t_i) = S(t_i) + U(t_i) \qquad i=1,...,N \qquad (2)$$

where

$$X(t_i) = col.[\, x_1(t_i), x_2(t_i),...,x_M(t_i)\,],$$

$$S(t_i) = col.[\, s_1(t_i), s_2(t_i),...,s_M(t_i)\,],$$

$$s_j(t_i) = \sum_{k=1}^{q} \alpha_k \sin\,[\omega_k t_i + (2\pi/\lambda)(j-1)\, d \sin\theta_k], \qquad i=1,...,N, \;\; j=1,...,M,$$

$$U(t_i) = col.[\, u_1(t_i), u_2(t_i),...,u_M(t_i)\,],$$

and

$$U(t_i) \sim N(0,\rho I)\,. \qquad (3)$$

The joint density function of the sampled data is given by

$$f\,[X(t_1),...,X(t_N);\Theta] = \frac{1}{(2\pi)^{MN/2}(\det[\rho I\,])^{N/2}} \exp\left\{ -\frac{1}{2\rho} \sum_{i=1}^{N} Y^T(t_i;\Theta) Y(t_i;\Theta) \right\} \qquad (4)$$

where

$$\Theta = col.[\theta_1, \ldots, \theta_q],$$

$$Y(t_i;\Theta) = col.[y_1(t_i;\Theta), ..., y_M(t_i;\Theta)],$$

and

$$y_j(t_i;\Theta) = x_j(t_i) - \sum_{k=1}^{q} \alpha_k \, sin\,[\omega_k t_i + \pi\,(j-1)\sin\theta_k]. \tag{5}$$

Note that the spacing d has been chosen to be exactly half of wavelength λ for simplicity. Thus, the log likelihood, ignoring constant terms, is given by

$$L(\Theta) = -\frac{MN}{2}\log\rho - \frac{1}{2\rho}\sum_{i=1}^{N} Y^T(t_i;\Theta)Y(t_i;\Theta). \tag{6}$$

To compute the maximum likelihood (ML) estimator we have to maximize the log likelihood with respect to Θ and ρ. Fixing the Θ, and maximizing with respect to ρ, we get

$$\hat{\rho} = \frac{1}{MN}\sum_{i=1}^{N} Y^T(t_i;\Theta)Y(t_i;\Theta). \tag{7}$$

Substituting this back to equation (6), ignoring constant terms, we can obtain the ML estimates of θ_k's by maximizing

$$-MN\log\left[\frac{1}{MN}\sum_{i=1}^{N} Y^T(t_i;\Theta)Y(t_i;\Theta)\right].$$

Since the logarithm is a monotonic function, the estimates can be obtained by

$$\min_{\Theta}\sum_{i=1}^{N} Y^T(t_i;\Theta)Y(t_i;\Theta).$$

This can be rewritten as

$$\min_{\Theta}\sum_{i=1}^{N}\sum_{j=1}^{M}\left\{x_j(t_i) - \sum_{k=1}^{q}\alpha_k \, sin\,[\omega_k t_i + \pi\,(j-1)\sin\theta_k]\right\}^2. \tag{8}$$

ROBUST DIRECTION-OF-ARRIVAL ESTIMATION

In the above formulation, the estimates obtained by equation (8) are ML estimates if the underlying noise distribution is exactly Gaussian. In practical situations, however, the commonly made assumption in statistics is at most approximation to reality. We define an *outlier* in a set of data to be an observation which appears to be inconsistent with the remainder of that set of data. It may not be hard to spot the potentially troublesome data point in the lower dimension, but it becomes exceedingly difficult with higher dimensions, or with multiparameter problems.

For this reason, minor deviations from the Gaussian noise are often modeled by the mixture model for noise [8]. One particular mixture model of interest is the slippage model with the Gaussian distribution as the dominant distribution. If $U(t_i)$ is a sequence of random variables obeying such a slippage model, then any $U(t_i)$ is distributed either as a Gaussian distribution $N(0,\sigma^2)$ with probability $(1-\varepsilon)$, or as an unknown distribution $Q(\mu, \beta\sigma^2)$ with probability ε, where μ and $\beta\sigma^2$ are the mean and the variance of the unknown distribution Q. In general, $\beta>1$, $0<\varepsilon< 0.1$ and, μ, an unknown constant, is of the order of a multiple of σ. The noise distribution can then be expressed as

$$P\{U(t_i)\} \sim (1-\varepsilon)N(0,\sigma^2)+\varepsilon Q(\mu,\, \beta\sigma^2), \tag{9}$$

which represents a family of distributions characterized by the mixing parameter ε. For $\varepsilon=0$, (9) reduces to a Gaussian distribution.

The parameter vector, Θ, is estimated by a robust technique based on the so called M-estimators, a generalization of classical maximum likelihood (ML) estimator by Huber [4]. In order to enforce scale invariance, the Θ and a scale parameter σ are estimated simultaneously, and this can be done by minimizing

$$J(\Theta,\sigma) = \sum_{i=1}^{N}\sum_{j=1}^{M} [\rho(y_j(t_i;\Theta)/\sigma)\,\sigma] + a\sigma\,, \qquad \sigma > 0 \tag{10}$$

with respect to Θ and σ, where $y_j(t_i;\Theta)$ and $\rho(\cdot)$ are defined by (5) and (12) respectively. The constant a, given by (15), is chosen to make the estimates consistent at the nominal distribution. Unless the minimum of $J(\Theta,\sigma)$ occurs on the boundary $\sigma=0$, it can be equivalently characterized by the $q+1$ equations

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \psi(y_j(t_i)/\sigma)\,[\partial y_j(t_i;\Theta)/\partial\theta_k] = 0; \qquad k=1,\ldots,q$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \chi(y_j(t_i;\Theta)/\sigma) = a\,, \tag{11}$$

where

$$\rho(x) = \begin{cases} x^2/2 & ,\ \text{if } |x| \le c \\ c|x| - c^2/2 & ,\ \text{if } |x| > c \end{cases} \tag{12}$$

$$\psi(x) = \partial\rho(x)/\partial x \tag{13}$$

$$\chi(x) = x\,\psi(x) - \rho(x), \tag{14}$$

$$a = MN \int_{-\infty}^{\infty} \chi(x)\,(1/\sqrt{2\pi})\exp[-x^2/2]\,dx. \tag{15}$$

and $y_j(t_i;\Theta)$ is as defined in equation (6). The constant c is related to the fraction of contamination ε by

$$2\Phi(c) - 1 + 2\phi(c)/c = 1/(1-\varepsilon), \tag{16}$$

where $\Phi(c)$ is the standard cumulative Gaussian distribution with zero mean and unit variance, and $\phi(c)$ is the corresponding Gaussian density. Usually c is chosen to lie between 1 and 2, which corresponds to the ε-interval [0.0083, 0.1428] by (16). We use the following algorithm to solve the $q+1$ equations of (11) since its convergence properties are well established [4].

Robust Estimation Procedure

Given $x_j(t_i)$'s $, i=1,..,N\,;\ j=1,..,M$, solve equation (11) for Θ and σ.

1. Choose starting values $m=0$, $\Theta^{(m)}$, $\sigma^{(m)}$, and a tolerance value $\zeta > 0$.
2. Compute residuals.

$$y_j(t_i;\Theta)^{(m)} = x_j(t_i) - \sum_{k=1}^{q} \alpha_k \sin[\omega_k t_i + \pi\,(j-1)\,\sin\theta_k{}^{(m)}] \ , \quad i=1,..,N\ ;\ j=1,..,M.$$

3. Compute a new value of $\sigma^{(m)}$.

$$[\sigma^{(m+1)}]^2 = (1/a) \sum_{i=1}^{N}\sum_{j=1}^{M} \chi(y_j(t_i;\Theta)^{(m)}/\sigma^{(m)})\,[\sigma^{(m)}]^2.$$

4. "Winsorize" the residuals.

$$Z = col.[z_{11},..,z_{1M},......,z_{N1},..,z_{NM}]$$

where

$$z_{ij} = \psi(y_j(t_i;\Theta)^{(m)}/\sigma^{(m+1)})\,\sigma^{(m+1)} \quad ; \quad i=1,..,N\ ;\ j=1,..,M.$$

5. Compute the partial derivatives.

$$b_{mk} = (\frac{\partial}{\partial\theta_k}) \left\{ \sum_{k=1}^{q} \alpha_k \sin\,[\omega_k t_i + \pi(j-1)\,\sin\theta_k] \right\} \ , \ m=1,...,NM\ ;\ k=1,...,q\ .$$

6. Solve $[B^T B]\tau = B^T Z$ for τ, where B is a NM by q matrix whose elements are b_{mk}'s.

7. Update $\Theta^{(m)}$ by $\Theta^{(m+1)} = \Theta^{(m)} + \xi\tau$, where $0 < \xi < 2$ is an arbitrary relaxation factor.

8. Stop iterating and go to step 9 if the parameters change by less than ζ times their standard deviation, where ζ is an arbitrarily chosen small value, i.e., if for all j,

$$|\tau_j| < \zeta\,[\bar{b}_{jj}]^{1/2}\,\sigma^{(m+1)}$$

$\bar{b}_{jj}$ is the jth diagonal element of the matrix $\bar{B} = (B^T B)^{-1}$; otherwise $m = m+1$ and go to step 2.

9. The final estimate of Θ_0 is given by $\Theta^{(m+1)}$, and the variance of the residual sequence by $[\sigma^{(m+1)}]^2$.

PERFORMANCE ANALYSIS

We first compute the Cramer Rao Lower Bound (CRLB) for a Gaussian noise case, and then for a contaminated Gaussian noise case to observe the changes in the CRLB with small changes in the Gaussian noise distribution. The variances of the robust estimates are then obtained from Influence Function (IF), which was introduced by Hampel [8] for investigating the infinitesimal behavior of real-valued functionals on a heuristic basis. Details of the derivation can be found in [7].

For the experiment we simulated two uncorrelated emitters impinging from 30° and 33°, with one hundred snapshots taken from eight equispaced linear array of sensors. The unknown distribution Q in equation (9) is assumed to be a zero mean Gaussian with ten times the variance of the dominant Gaussian. The percentage of outliers, ε, is chosen to be five. The amplitudes and frequencies of the envelopes associated with the sources are chosen to be $\alpha_1=1$, $\alpha_2=1$, $\omega_1 = 10\pi$, and $\omega_2 = 20\pi$.

Cramer Rao Lower Bound (CRLB) for *Gaussian Noise Case*

Let us denote

$$A_1 = \frac{1}{(2\pi)^{M/2}(\det[\rho I])^{N/2}}$$

and

$$A_2(t_i;\Theta) = -\frac{1}{2\rho}\sum_{j=1}^{M}\left\{ x_j(t_i) - \sum_{k=1}^{q}\alpha_k \sin\left[\omega_k t_i + \pi\,(j-1)\sin\theta_k\right] \right\}^2 . \tag{17}$$

Then the density function can be written as

$$f\,[X(t_1),\ldots,X(t_N);\Theta] = \prod_{i=1}^{N} f\,[X(t_i);\Theta], \tag{18}$$

where

$$f\,[X(t_i);\Theta] = A_1 \exp\,[A_2(t_i;\Theta)].$$

Therefore,

$$\begin{aligned}\frac{\partial}{\partial\Theta}L(\Theta) &= \frac{\partial}{\partial\Theta}\log\left\{ f\,[X(t_1),\ldots,X(t_N);\Theta] \right\} \\ &= \sum_{i=1}^{N}\frac{\partial}{\partial\Theta}\log\left\{ f\,[X(t_i);\Theta] \right\} \\ &= \sum_{i=1}^{N}\left(1/f\,[X(t_i);\Theta]\right)\frac{\partial}{\partial\Theta}f\,[X(t_i);\Theta] \\ &= \sum_{i=1}^{N}\left\{ \frac{\partial}{\partial\Theta}A_2(t_i;\Theta) \right\}. \end{aligned} \tag{19}$$

Then the Fisher Information Matrix can be computed as

$$I(\Theta) = \int \left[\frac{\partial}{\partial\Theta}L(\Theta)\right]\left[\frac{\partial}{\partial\Theta}L(\Theta)\right]^T dF\ , \tag{20}$$

and the covariance matrix of the parameter vector Θ is given by

$$Cov(\Theta) = [I(\Theta)]^{-1}\ . \tag{21}$$

Cramer Rao Lower Bound (CRLB) for *Contaminated Gaussian Noise Case*

Since the distribution Q in equation (9) is assumed to be a zero mean Gaussian with η times the variance of the dominant Gaussian,

$$\begin{aligned} f\,[X(t_i);\Theta] &= (1-\varepsilon)A_1 \exp[\,A_2(t_i;\Theta)] + \varepsilon\frac{A_1}{\eta^{M/2}}\exp[\,A_2(t_i;\Theta)/\eta\,] \\ &= A_1 \exp\,[A_2(t_i;\Theta)]\,A_3(t_i;\Theta), \end{aligned} \tag{22}$$

where A_1, $A_2(t_i;\Theta)$ are defined in equation (17), and

$$A_3(t_i;\Theta)=1-\varepsilon+\varepsilon\eta^{-\frac{M}{2}}\, exp\,[(\frac{1}{\eta}-1)A_2(t_i;\Theta)]. \tag{23}$$

Hence

$$\frac{\partial}{\partial\Theta}L(\Theta)=\sum_{i=1}^{N}\frac{\partial}{\partial\Theta}\log\left\{f\,[X(t_i);\Theta]\right\}$$

$$=\sum_{i=1}^{N}\,[\left\{\frac{\partial}{\partial\Theta}A_2(t_i;\Theta)\right\}+\left\{\frac{\partial}{\partial\Theta}A_3(t_i;\Theta)\right\}/\,A_3(t_i;\Theta)\,] \tag{24}$$

The Fisher Information Matrix and the covariance matrix of the parameter vector thus can be computed by (20) and (21), respectively. Figure 1 shows the plot of CRLB's *vs.* SNR for a pure Gaussian and a contaminated Gaussian noise case. Note the approximately fifty percent increase in the CRLB caused by replacing five percent of the Gaussian data with outliers that has ten times the variance of the parent Gaussian distribution.

Variance of the Robust Estimates

The expression for the variance of robust estimates can be obtained using Influence Function (IF), where the details of the derivation can be found in [7]. In our case, the covariance matrix of the robust DOA estimates are given by

$$Var\,(\hat{\Theta})=\frac{ave\,[\psi^2(r\,(i,j;\hat{\Theta}))]}{\left\{ave\,[\psi'(r\,(i,j;\hat{\Theta}))\,r\,(i,j;\hat{\Theta})]\right\}^2}\left\{\sum_{i=1}^{N}\sum_{j=1}^{M}V\,(i,j;\hat{\Theta})\,V^T(i,j;\hat{\Theta})\right\}^{-1} \tag{25}$$

where

$$r\,(i,j;\hat{\Theta})=y_j(t_i;\Theta)/\hat{\sigma},$$

$$V\,(i,j;\hat{\Theta})=-(\partial/\partial\Theta)r\,(i,j;\hat{\Theta}),$$

and

$$\psi'(x)=(\partial/\partial x)\,\psi(x).$$

$y_j(t_i)$ and $\psi(x)$ are defined in (5) and (13), respectively. Here, $\hat{\Theta}$ and $\hat{\sigma}$ indicates the estimated value of Θ and σ.

Figure 2 shows the the theoretical RMSE (sum of bias square and variance) *vs.* SNR plot of the ML and robust estimates when the underlying noise distribution is purely Gaussian. Note that the performance of the robust estimator is almost as good as that of the ML estimator in the Gaussian noise. On the other hand, Figure 3 shows the theoretical RMSE *vs.* SNR plot of the ML and the robust estimates after replacing five percent of the Gaussian data with outliers that has ten times the variance of the original Gaussian distribution. Note that the theoretical RMSE of the robust estimates are much smaller than those of the ML estimates in the presence of outliers.

Table 1 shows the twenty-run averages of $\hat{\theta}_1$, $\hat{\theta}_2$, and the corresponding RMSE's at many different SNR's for the contaminated Gaussian noise case. Less than ten iterations were required for either the ML or the robust estimation before convergence was achieved.

CONCLUSIONS

We have presented a robust direction-of-arrival (DOA) estimation algorithm which performs almost as well as a maximum likelihood (ML) DOA estimation scheme in the pure Gaussian noise, and much better in the presence of outliers. Although not demonstrated here, the technique is equally applicable to single snapshot cases and coherent signals. Computational experience has also shown that the cost function may have many local minima, especially when the SNR is low. Thus for SNR's less than about 0dB, in spite of using sophisticated optimization techniques, the minimization algorithm do not always converge to the global minimum. One can always obtain the global minimum by repeating the algorithm with several different starting points and choosing the one with the least value.

Table 1. *Contaminated Gaussian Noise Case*: Twenty-run averages of θ_1, θ_2, and the RMSE's (shown in parentheses) for different values of SNR. The Gaussian ML DOA estimates and the robust DOA estimates are shown. The contamination is caused by replacing five percent of the Gaussian data with outliers which has ten times the variance of the parent Gaussian distribution.

Average of the Estimates (& RMSE)				
	Gaussian ML		robust	
SNR(dB)	θ_1	θ_2	θ_1	θ_2
21.0	29.9999 (0.0361)	32.9815 (0.0369)	29.9989 (0.0206)	32.9876 (0.0191)
18.0	29.9948 (0.0588)	33.0104 (0.0501)	29.9911 (0.0292)	33.0017 (0.0243)
15.0	29.9630 (0.0983)	32.9521 (0.0634)	29.9917 (0.0465)	32.9925 (0.0327)
12.0	30.0321 (0.1124)	32.9800 (0.1186)	30.0056 (0.0478)	32.9923 (0.0508)
9.0	29.9090 (0.1624)	33.0654 (0.1718)	29.9665 (0.0864)	33.0143 (0.0640)
6.0	30.0149 (0.2135)	32.9667 (0.2125)	30.0035 (0.1093)	32.9767 (0.0951)
3.0	30.0328 (0.3311)	33.0701 (0.4286)	30.0086 (0.1641)	33.0092 (0.1797)
0.0	29.9624 (0.4844)	33.0048 (0.3854)	30.0084 (0.1961)	33.0521 (0.1982)
-3.0	30.3364 (0.5868)	32.9346 (0.5440)	30.1757 (0.2838)	32.9839 (0.2833)
-6.0	29.8573 (0.8512)	32.6346 (0.9258)	29.9833 (0.4219)	32.8909 (0.4585)
-9.0	29.7591 (1.1717)	32.8647 (1.3988)	30.0571 (0.5535)	31.9184 (0.7358)

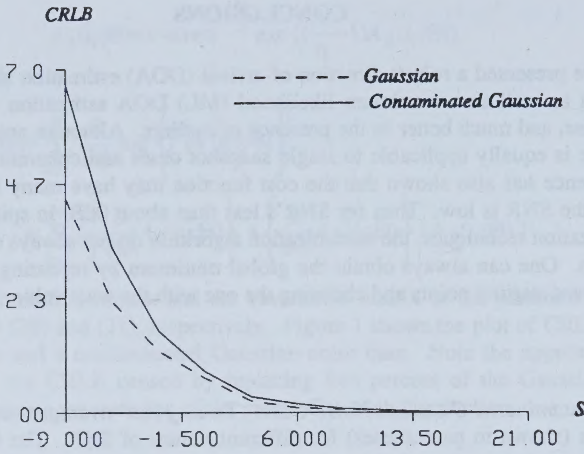


Figure 1. *CRLB vs. SNR* for a Gaussian Noise Case (solid line) and for a Contaminated Gaussian Noise Case (dotted line). The contamination is caused by replacing five percent of the Gaussian data with outliers which has ten times the variance of the parent Gaussian distribution.

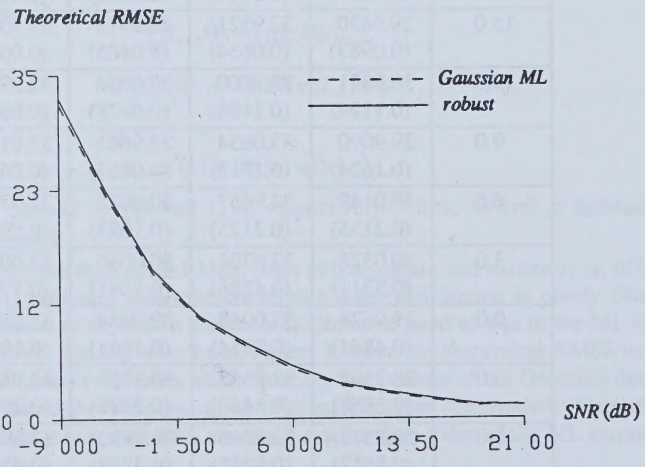


Figure 2. *Gaussian Noise Case*: Theoretical RMSE vs. SNR for the Gaussian ML DOA estimates (dotted line) and for the robust DOA estimates (solid line).

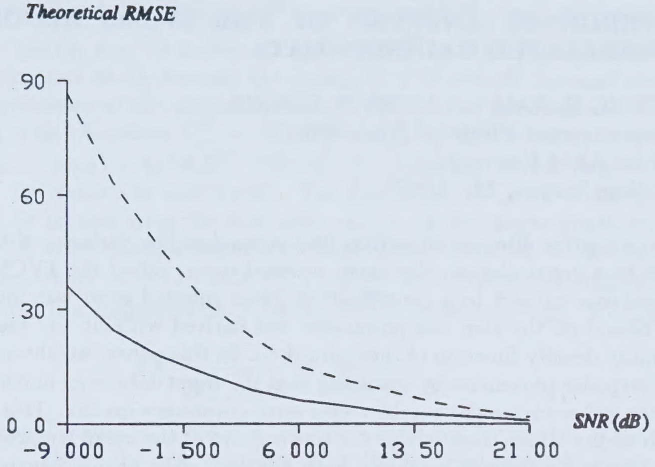


Figure 3. *Contaminated Gaussian Noise Case*: Theoretical RMSE vs. SNR for the Gaussian ML DOA estimates (dotted line) and for the robust DOA estimates (solid line). The contamination is caused by replacing five percent of the Gaussian data with outliers which has ten times the variance of the parent Gaussian distribution.

LIST OF REFERENCES

[1] W. S. Liggett, "Passive Sonar: Fitting Models to Multiple Time Series," in *Signal processing*, J. W. R. Griffith et al., Eds., New York: Academic, 1973.

[2]. I. Ziskind and M. Wax, "Maximum Likelihood Localization of Multiple Sources by Alternating Projection", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp.1553-1560, Oct. 1988.

[3] T. J. Shan, M. Wax and T. Kailath, "On Spatial Smoothing for Direction-of-Arrival Estimation of Coherent Sources," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp.806-811, 1985.

[4]. P. J. Huber, *Robust Statistics*, John Wiley and Sons, pp. 153-191, 1981.

[5] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust Statistics: The Approach Based on Influence Functions*, John Wiley & Sons, 1986.

[6]. David D. Lee and R. L. Kashyap, "Robust Decentralized Direction-of-Arrival Estimation in Contaminated Noise," *(to appear in the IEEE Trans. on ASSP)*.

[7] Sang G. Oh and R. L. Kashyap, "Robust Approach for High Resolution Frequency Estimation," *(Submitted to the IEEE Trans. on ASSP for Publication)*.

[8] J. W. Tukey, A Survey of Sampling from Contaminated Distributions, *in : Contributions to Probability and Statistics*, I.Olkin, Ed., Stanford University Press, 1960.

CONVERGENCE ANALYSIS OF THE LVCMS ALGORITHM FOR UNCORRELATED GAUSSIAN DATA

SEUNG H. NAM and JERRY D. GIBSON
Department of Electrical Engineering
Texas A&M University
College Station, TX 77843

An adaptive filtering algorithm that minimizes the variance of the squared error subject to a constraint on the mean squared error, called the LVCMS algorithm for least variance subject to a constraint on mean squared error was introduced in [1,2], and a bound on the step size parameter was derived without any assumptions on the probability density function of the input data. In this paper, we obtain a tighter bound on the step size parameter by assuming that the *input* data is an uncorrelated Gaussian sequence and studying the weight vector error covariance matrix. This new bound helps explain earlier simulations where the convergence of the mean squared coefficient error vector was more difficult to obtain with additive noise having heavier tailed densities [2].

The coefficient adaptation rule of the LVCMS algorithm is given by

$$\mathbf{W}_{j+1} = \mathbf{W}_j + 4\mu\epsilon_j^3\mathbf{X}_j - 2\mu\left(2\sigma_0^2+\lambda\right)\epsilon_j\mathbf{X}_j \qquad (1)$$

where $\mathbf{W}_j^T = [w_{1j}\ w_{2j} \dots w_{Nj}]$ is the vector of adaptive filter coefficients at time instant j, $\mathbf{X}_j^T = [x_j\ x_{j-1} \dots x_{j-N+1}]$ is the N–vector of the past inputs, ϵ_j is the error signal at time j, σ_0^2 is the mean squared error (MSE) constraint, λ is the Lagrange multiplier, and μ is the step size parameter. Using standard arguments, it is shown in [1,2] that the mean coefficient vector converges to the optimal solution, $\mathbf{W}_0$, if

$$0 < \mu < \frac{1}{\left[6\,E(n_j^2) - 2\sigma_0^2 - \lambda\right]\, tr\,\mathbf{R}} \qquad (2)$$

where n_j is the plant noise and $\mathbf{R} = E\left(\mathbf{X}_j\mathbf{X}_j^T\right)$. The step size parameter μ is usually chosen much smaller than the upper bound to ensure the convergence of the coefficient vector. This implies that the bound in (2) is not sufficiently tight.

We derive a new bound on μ by imposing the additional assumption that the input data is an uncorrelated Gaussian sequence. With $\mathbf{V}_j = \mathbf{W}_j - \mathbf{W}_0$, we can write [2]

$$\mathbf{V}_{j+1} = \mathbf{V}_j + 4\mu\mathbf{X}_j\sum_{i=0}^{3}\binom{3}{i}n_j^i\left(-\mathbf{X}_j^T\mathbf{V}_j\right)^{3-i} - 2\mu\alpha\left(n_j - \mathbf{X}_j^T\mathbf{V}_j\right)\mathbf{X}_j \qquad (3)$$

where $\alpha \triangleq 2\sigma_0^2 + \lambda$. Defining $\mathbf{U}$ as a unitary matrix such that $E\left\{\mathbf{U}\mathbf{X}_j\mathbf{X}_j^T\mathbf{U}^T\right\} = \operatorname{diag}(\gamma_1\ \gamma_2 \dots \gamma_N) = \mathbf{\Lambda}$, using the Gaussian fourth moment expansion and the assumption that the noise is zero mean and white with a symmetric probability density function, and assuming that $\mathbf{W}_j$ is very close to $\mathbf{W}_o$ so that we can neglect terms higher than second order in $\mathbf{V}_j$, we find (after much algebra) that $\mathbf{C}_j = E\{\mathbf{U}\mathbf{V}_j\mathbf{V}_j^T\mathbf{U}^T\}$ has the recursion

$$\mathbf{C}_{j+1} = \mathbf{C}_j - 2\mu a\left[\mathbf{\Lambda}\mathbf{C}_j + \mathbf{C}_j\mathbf{\Lambda}\right] + 4\mu^2 b\left[2\mathbf{\Lambda}\mathbf{C}_j\mathbf{\Lambda} + tr\left(\mathbf{\Lambda}\mathbf{C}_j\right)\mathbf{\Lambda}\right] + 4\mu^2 c\mathbf{\Lambda} \qquad (4)$$

where $a = 6E(n_j^2) - \alpha, b = 60E(n_j^4) - 24\alpha E(n_j^2) + \alpha^2$, and $c = \alpha^2 E(n_j^2) - 4\alpha E(n_j^4) + 4E(n_j^6)$.

Following Feuer and Weinstein's LMS work [3], we decompose the recursion in

(4) into $C_{j+1}^{ii} = \rho_{ii} C_j^{ii} + 4\mu^2 b\gamma_i \sum_{p=1}^{N} \gamma_p C_j^{pp} + 4\mu^2 c\gamma_i$ and $C_{j+1}^{ik} = \rho_{ik} C_j^{ik}$, for $i \neq k$, where $\rho_{ik} = 1 - 2\mu a(\gamma_i + \gamma_k) + 8\mu^2 b\gamma_i\gamma_k$. Since $\left(C_j^{ik}\right)^2 \leq C_j^{ii} C_j^{kk}$, the convergence of the diagonal elements of $\mathbf{C}_j$ ensures the convergence of the off–diagonal elements. To study the convergence of the diagonal elements, we rewrite the diagonal recursion as $\bar{\mathbf{C}}_{j+1} = \mathbf{F}\bar{\mathbf{C}}_j + 4\mu^2 c\mathbf{L}$ where $\bar{\mathbf{C}}_j^T = \left[C_j^{11} C_j^{22} \ldots C_j^{NN}\right]$, $\mathbf{L}^T = [\gamma_1\ \gamma_2 \cdots \gamma_N]$, and $\mathbf{F} = \operatorname{diag}(\rho_{11}\rho_{22} \ldots \rho_{NN}) + 4\mu^2 b\mathbf{L}\mathbf{L}^T$. Clearly, $\bar{\mathbf{C}}_j$ converges if and only if all the eigenvalues of $\mathbf{F}$ lie inside the unit circle. The conditions on μ for convergence of $\bar{\mathbf{C}}_j$ can be found by investigating the pole–zero pattern of the characteristic equation $\det(\mathbf{F} - \beta\mathbf{I}) = 0$ [3] or by using a theorem of Gantmacher as in [4]. Either approach gives the same conditions: $\mu < \frac{a}{2b\gamma_i}$ and $\sum_{i=1}^{N} \frac{\mu b\gamma_i}{a - 2\mu b\gamma_i} < 1$. To get an explicit bound on μ, we equate this last expression to 1 and follow the approach of [3]. We then find the new bound

$$\mu < \frac{a}{3b\sum_{i=1}^{N} \gamma_i} = \frac{a}{3btr(\mathbf{R})}. \tag{5}$$

Note that the original bound in (2) can be rewritten using the definitions of a and b as $0 < \mu < [atr(\mathbf{R})]^{-1}$. Hence the new bound in (5) is tighter than the original one in (2) when $b > a^2$. Since

$$b = \begin{cases} a^2 + 12aE\left(n_j^2\right) + 252\left[E\left(n_j^2\right)\right]^2 & \text{for Laplacian } n_j \\ a^2 + 12aE\left(n_j^2\right) + 72\left[E\left(n_j^2\right)\right]^2 & \text{for Gaussian } n_j \\ a^2 + 12aE\left(n_j^2\right) & \text{for uniform } n_j, \end{cases}$$

we see that the bound in (5) is tighter than (2) for Laplacian, Gaussian, and uniform additive noise. Further, these expressions for b, when used in (5), explain earlier simulation results of the LVCMS algorithm indicating that the mean squared coefficient error convergence is more difficult to obtain with heavier tailed additive noise densities [2].

REFERENCES

[1] J. D. Gibson and S. D. Gray, "Adaptive Filtering for a Localized Distortion Measure," Proceedings, Twenty–Fourth Annual Allerton Conference on Commun., Control, and Computing, Oct. 1–3, 1986, Monticello, IL, pp. 113–114.

[2] J. D. Gibson and S. D. Gray, "MVSE adaptive filtering subject to a constraint on MSE," IEEE Trans. Circuits and Systems, vol. 35, pp. 603–608, May 1988.

[3] A. Feuer and E. Weinstein, "Convergence analysis of LMS filters with uncorrelated Gaussian data," IEEE Trans. Acoust., Speech, and Signal Processing, vol. ASSP–33, pp. 222–230, Feb. 1985.

[4] J. B. Foley and F. M. Boland, "A note on the convergence analysis of LMS adaptive filters with Gaussian data," IEEE Trans. Acoust., Speech, and Signal Processing, vol. ASSP–36, pp. 1087–1089, July 1988.

FIXED-POINT IMPLEMENTATION OF A LEAST SQUARES EQUALIZER USING HIGHLY QUANTIZED COEFFICIENTS

S. H. LEUNG and B. L. CHAN
Department of Electronic Engineering, City Polytechnic of Hong Kong
83 Tat Chee Avenue, Kowloon Tong, Hong Kong.

ABSTRACT

This paper is to present a fixed-point implementation of the adaptive equalizer using unnormalized least-squares lattice-ladder algorithm [1]. In order to prevent overflows due to the fixed-point arithmetic with finite-wordlength, we introduce adaptive scaling to the internal summing nodes. Furthermore, the scaling technique can also improve the dynamic range of the small signals.

In the implementation, we introduce a quantizer to one input of the multiplier and a quantizer to the output of the divider so that the multiplication and division can be rendered to simple shift-add and shift-subtract operations, respectively.

I. ADAPTIVE SCALING

The conventional unnormalized lattice-ladder algorithm comprises a lattice part and a ladder part for computing the prediction errors and the estimate, respectively [1].

The lattice part of the adaptive filter is described mainly by the reflection coefficients, $\mathcal{K}^{e}_{p+1,T}$ and $\mathcal{K}^{r}_{p+1,T}$. These coefficients are computed as the ratio of the cross-correlation $\mathscr{k}_{p+1,T}$ and the error covariances $\mathcal{R}^{e}_{p,T}$, $\mathcal{R}^{r}_{p,T}$. These parameters are recursively updated by,

$$\mathscr{k}_{p+1,T} = \lambda\, \mathscr{k}_{p+1,T-1} + e_{p,T}\, e^{o}_{p,T}$$

$$\mathcal{R}^{e}_{p,T} = \lambda\, \mathcal{R}^{e}_{p,T-1} + e^{o}_{p,T}\, e_{p,T}, \quad \mathcal{R}^{r}_{p,T} = \lambda\, \mathcal{R}^{r}_{p,T-1} + r^{o}_{p,T}\, r_{p,T} \tag{1}$$

where p and T denote the order and the time, respectively.

Now we introduce adaptive scaling in the algorithm in order to control the dynamic range of the signals and to prevent the arithmetic overflow. The idea is to multiply a time-varying parameter whose value is of power-of-two to the error covariances and the cross-correlation, so that the small magnitude is scaled-up while the large value is scaled-down.

Let us define $k_{p+1,T}$, $R^{e}_{p,T}$, and $R^{r}_{p,T}$ as the scaled cross-correlation and error covariances, respectively. They are scaled by the function $s_{p,T}$ in the following equations,

$$k_{p+1,T} = s_{p,T}\, \mathscr{k}_{p+1,T}, \quad R^{e}_{p,T} = s_{p,T}\, \mathcal{R}^{e}_{p,T}, \quad R^{r}_{p,T} = s_{p,T}\, \mathcal{R}^{r}_{p,T}\ . \tag{2}$$

Using the equations, the scaled cross-correlation is updated by

$$k_{p+1,T} = u_{p,T}\, v^{k}_{p+1,T}$$

where

$$v^{k}_{p+1,T} = \lambda\, k_{p+1,T} + s_{p,T-1}\, e_{p,T}\, r^{o}_{p,T-1}. \tag{3}$$

Similarly the error covariances are computed recursively by

$$\begin{aligned} R^{e}_{p,T} &= u_{p,T}\, v^{e}_{p,T}\ , \quad R^{r}_{p,T} = u_{p,T}\, v^{r}_{p,T} \\ v^{e}_{p,T} &= \lambda\, R^{e}_{p,T-1} + s_{p,T-1}\, e^{o}_{p,T}\, e_{p,T} \\ v^{r}_{p,T} &= \lambda\, R^{r}_{p,T-1} + s_{p,T-1}\, r^{o}_{p,T}\, r_{p,T}\ , \end{aligned} \qquad u_{p,T} = \begin{cases} 2 & M<0.25 \\ 0.5 & M\geq 1.0 \\ 1.0 & \text{otherwise.} \end{cases} \tag{4}$$

In the normal operation, the magnitude of the cross-correlation is bounded by the value of the error covariance. Then we can simply use the error covariances to determine the scaling parameter $s_{p,T}$. Let us define M as the maximum of $v^{e}_{p,T}$ and $v^{r}_{p,T}$. Then we choose the parameter $u_{p,T}$ to scale the values of $v^{e}_{p,T}$ and $v^{r}_{p,T}$ so that the resulting $R^{e}_{p,T}$ and $R^{r}_{p,T}$ are "well" bounded. A simple rule for defining $u_{p,T}$ is suggested in (4). The scaling parameter $s_{p,T}$ is thus obtained by

$$s_{p,T} = u_{p,T}\, s_{p,T-1}\ . \tag{5}$$

By this method of scaling, we control the magnitude of the error covariances bounded between 1 and 0.25 so that the overflow is prevented and the small magnitude is scaled-up. As a result, the reflection coefficients are updated as

$$K^{e}_{p+1,T} = k_{p+1,T} / R^{e}_{p,T} \tag{6}$$

$$K^{r}_{p+1,T} = v^{k}_{p+1,T} / R^{r}_{p,T-1} \quad . \tag{7}$$

In the laddar part, the ladder gain $\mathcal{K}_{p+1,T}$, the cross-correlation $q_{p+1,T}$ and the estimation error $\mathcal{J}_{p+1,T}$ are given by the following formulae,

$$q_{p+1,T} = \lambda\, q_{p+1,T} + \mathcal{J}_{p,T}\, \overset{o}{r}_{p,T}, \quad \mathcal{J}_{p+1,T} = \mathcal{J}_{p,T} - \mathcal{K}_{p+1,T}\, r_{p,T}. \tag{8}$$

Let $g_{p+1,T}$ denote the scaled version of $q_{p+1,T}$, i.e.

$$g_{p+1,T} = s^{g}_{p,T}\, s_{p,T}\, q_{p+1,T} \ , \tag{9}$$

where $s^{g}_{p,T}$ is used to scale $q_{p+1,T}$ so that the resulting magnitude of $g_{p+1,T}$ is less than $R^{r}_{p,T}$. By proceeding the same way, we obtain the formula for the scaled cross-correlation,

$$g_{p+1,T} = u^{g}_{p,T}\, u_{p,T}\, v^{g}_{p+1,T}$$

where

$$v^{g}_{p+1,T} = \lambda\, g_{p+1,T-1} + s^{g}_{p,T-1}\, s_{p,T-1}\, J_{p,T}\, \overset{o}{r}_{p,T} \ . \tag{10}$$

The parameter $u^{g}_{p,T}$ is defined in the same way as (4) with M replaced by the absolute value of $(v^{g}_{p+1,T} / R^{r}_{p,T})$. The values of $s^{g}_{p,T}$ is updated following (5). The scaled ladder gain $K_{p+1,T}$ and the estimation error $J_{p+1,T}$ are then computed in the following equations,

$$K_{p+1,T} = g_{p+1,T} / R^{r}_{p,T}, \quad J_{p+1,T} = J_{p,T} - K_{p+1,T}\, J_{p,T}\, \overset{o}{r}_{p,T} / s^{g}_{p,T}. \tag{11}$$

The new lattice-ladder algorithm provides a simple procedure to scale the parameters independent of the statistical properties of the signals (eigenvalue-spread), so that the numerical sensitivity due to the wide range of the error covariance magnitudes can be subtantially improved.

III. HIGHLY QUANTIZED COEFFICIENTS

Due to the low numerical sensitivity, we can simplify the hardware for implementing the lattice-ladder algorithm by introducing a low bit-dimension quantizer to certain signal points. The advantages of this approach are to reduce the multiplication and division to shift-add operation and shift-subtract operation, respectively.

In the algorithm, a quantizer is placed at one input of the multuplier and also at the output of the divider. Because of the low bit-dimension of the quantizer, the multiplication and division can be implemented by simple shift-add and shift-subtract operations. The new algorithm provides a means to implement a fast adaptive equalizer at low cost without implementing multiplier and divider.

IV. CONCLUSION

The equalization problem as in [2] is used to illustrate the performance of the new equalizer. We use 2-nonzero-bit and 3-nonzero-bit quantizers in the adaptive algorithm. Different wordlengths have also been used to illustrate the coefficient sensitivity. The results show that the performance of the new equalizer with the wordlength of 16-bit are comparable to the implementation using infinite precision arithmetic. Even using 3-nonzero-bit quantizer, the performance is almost the same as the infinite precision case.

The simulation results have shown that the technique of adaptive scaling does improve the signal to noise (quantization noise) ratio substantially, and the lattice-ladder algorithm is so insensitive to the quantization of the equalizer coefficients.

V. REFERENCES

[1] E.Shichor,"Fast Recursive Estimation Using The Lattice Structure",Bell Syst. Tech. J., vol.61, no.1, Jan. 1982.

[2] F. Satorius and J. D. Pack,"Applications of LS Adaptive Lattice Algorithms to Adaptive Equalization," IEEE Trans. Commun., vol.29,pp.136-142, Feb. 1981.

INDEX OF AUTHORS

Abu-Amara, H. 596
Acharya, R. 991
Akinniyi, A. R. 1
Akl, S. G. 748
Altman, E. 253
Amanullah, A. S. 27
Antonini, C. D. 441
Antsaklis, P. J. 915
1037
Arnautovic, S. H. 179
Asbjornsen, O. A. 199
Atkin, G. E. 25
Baccelli, F. 808
Bagchi, A. 613
Bai, S. 209
Baker, C. R. 1103
Barmish, B. R. 128
Barsky, M. F. 895
Bingulac, S. 439
Blair, J. R. S. 72
Blais, N. 1094
Bouthellier, P. R. 858
Brady, M. L. 798
Brandt, R. D. 491
Brochin, F. M. 364
Browning, D. W. 29
Bui, T. N. 92
Byrnes, C. I. 150
Calvert, K. L. 481
Casavant, T. L. 181
Chaiken, S. 555
Chan, B. L. 1174
Chang, C.-K. 1074
Chakravarty, S. 730
Chen, B. M. 439
Chen, J. 118
Chen, L. 756
Chen, T. 955
Chen, Y.-M. 548
Cheng, C. 594
Cheng, R. S. 1112
Cheong, W. H. 181
Cho, S. 724
Choi, H.-A. 102
Coan, B. A. 1062
Condamoor, R. 1072
Corless, M. 877
Corrales, H. P. 25
Costello, Jr., D. J. 312
Cox, N. R. 1005
Coyle, E. J. 374
981
Cruz, J. R. 1142
Dadoun, N. 538
Dawson, D. M. 229
Dehne, F. 1084
Dekel, E. 788
Delchamps, D. F. 965
Dhall, S. K. 68
776
Dorsey, J. F. 229
886
Douglass, B. G. 640
Duchon, N. 407
Economides, A. A. 849
Efstratiadis, S. N. 1007
ElGindy, H. 1094
ElMusrati. A. K. 461
Eom, K.-B. 993
Erickson, K. T. 233
243
Ferreira, A. 1084
Francis, B. A. 955
Freudenberg, J. S. 118
138
Frey, M. R. 1122
Fujisawa, T. 82
Furth, W. 407
Gabriel, L. 282
Gajic, Z. 148
Garg, V. 501
Gelfand, S. B. 1047
Georghiades, C. N. 13
Georgiadis, L. 810
Gewali, L. 584
Ghosh, B. K. 858
Gibson, J. D. 1172
Göknar, I. C. 471
Golaszewski, C. H. 520
Gray, D. L. 1049
Guo, H. 1047
Gupta, L. 1060
Güzelis, C. 471
Hached, M. 913
Hakimi, S. L. 613
Hall, E. B. 692
Heck, B. S. 903
Hemami, H. 169

Herro, M. A. 322
Hizlan, M. 11
Ho, S. 712
Holman, T. 712
Hong, C.-K. 385
Hopkinson, T. M. 336
Houshangi, N. 231
Hsu, W.-L. 291
Huang, N. Q. 925
Huang, Q. 1051
Huang, S.-H. S. 273
Huang, Y. F. 1132
Hughes, B. 11
39
Huynh, D. T. 724
Ioannou, P. A. 849
Isidori, A. 150
Jackson, A. 284
Juang, J.-N. 867
Kashiwabara, T. 82
Kashyap, R. L. 1162
Katsaggelos, A. K. 1007
Kaufmann, M. 301
Kim, H. 68
Kim, S. K. 766
Kim, Y. 1152
Kirkpatrick, D. G. 538
Koivo, A. J. 179
231
Kokkos, A. 355
Koppelman, D. M. 631
Kumar, R. 501
Kumar, S. P. R. 594
Kurose, J. F. 839
Lakshmivarahan, S. 68
776
Lam, S. S. 481
Le, V. B. 564
Lee, D. D. 1162
Lee, D. T. 564
Lee, S.-L. 92
Lee, S. M. 29
Lehnert, J. S. 1
Lei, C.-L. 1074
Letaief, K. B. 324
Leung, S. H. 1174
Li, Y. 528
925
Liestman, A. L. 621
Lin, F. 491
Lindner, D. K. 895
Liou, K.-P. 92
Liu, R. 1132
Liu, R. W. 1051
Liu, Z. 808
Longman, R. W. 867
Luccio, F. 702
Luginbuhl, T. E. 653
Madani-Esfahani, M. 913
Marcus, S. I. 501
935
Marvasti, F. 977
Masuda, S. 82
McEliece, R. J. 334
346
McGillem, C. D. 15
Meijer, H. 748
Mellichamp, D. A. 429
Michel, A. N. 159
451
1027
1049
Milito, R. A. 820
Moellenhoff, M. S. 233
Moore, K. C. 336
Morales, A. 991
Moulin, P. 663
Murthy, G. R. 374
Nakajima, K. 82
Nam, S. H. 1172
Navathe, S. B. 603
Nielsen, P. A. 682
Nikolaou, C. 810
Nowack, J. M. 322
Ntafos, S. 584
O'Reilly, J. J. 355
O'Sullivan, J. A. 663
O'Young, S. D. 945
Okada, M. 740
Ong, H. 169
Onyszchuk, I. M. 334
Orchard, M. 1003
Oruç, A. Y. 631
640
Özgüner, Ü. 421
Pagli, L. 702
Park, J. 993
Passino, K. M. 915
1037
Perez, L. C. 312
Perl, Y. 282
Phan, M. 867
Pippenger, N. 1017
Popovic, L. 346
Pujara, L. R. 136
Qu, Z. 886

Ramadge, P. J. 520
Ramanan, P. 284
548
Raney, A. D. 243
Rau-Chaplin, A. 1084
Ravi, S. S. 72
555
Rego, V. 383
Revesz, P. 778
Rhee, C. 68
776
Richards, D. 621
Rivera-Vega, P. I. 603
Saberi, A. 439
Sadowsky, J. S. 324
Sain, M. K. 104
159
Sain, P. M. 159
Sandberg, I. W. 111
Sartori, M. A. 1037
Sasaki, G. 397
Sayeh, M. R. 1060
Schaper, C. D. 429
Schmeichel, E. F. 613
Schoenwald, D. A. 421
Schulz, T. J. 651
Seborg, D. E. 429
Sernik, E. L. 935
Shanbhag, N. R. 136
Shen, X. 148
Sheu, M.-J. 1074
Shyu, C.-H. 788
Si, J. 1027
Silvester, J. A. 849
Simha, R. 839
Simon, S. 169
Skelton, R. E. 877
975
Snyder, D. L. 651
663
Snyder, L. 712
Soon, V. C. 1132
Stallmann, M. F. M. 58
Steele, A. 740
Sullivan, B. J. 979
Sun, H. F. 451
Szpankowski, W. 263
Tamassia, R. 48
Tammana, R. 1060
Tempo, R. 128
Thomas, J. B. 364
682
Todd, R. M. 1142
Tollis, I. G. 48
301
Tong, L. 1132
Towsley, D. 808
Tsai, K. H. 291
Tugnait, J. K. 672
Vaccaro, J. J. 336
Varadarajan, R. 603
Venkatesan, S. 1072
Verdu, S. 1112
Viniotis, I. 810
Viswanathan, V. 273
Wang, C. A. 574
Wang, I. Y. 385
Warren, D. 649
Weber, C. L. 1152
Wee, Y. C. 555
Welch, J. L. 1062
Wey, S.-J. 395
Wilkins, D. F. 1005
Willett, P. K. 649
653
Williamson, D. 975
Wise, G. 692
Wonham, W. M. 491
511
528
925
945
Wyman, B. F. 104
Yaz, E. 191
884
Yee, J. R. 829
Yegani, P. 15
Yen, G. 1027
Yen, H.-C. 714
Yesha, Y. 756
Yeung, R. W. 800
Yin, G. 199
219
Yin, K. 199
Yu, P.-T. 981
Zaghloul, M. E. 461
Zak, S. H. 913
Zheng, S.-Q. 301
Zhong, H. 511
Zhu, G. 877
975

UNIVERSITY OF ILLINOIS-URBANA
621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1989
3 0112 008149269